IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

On the (in)Security of the Control Plane of SDN Architecture: A Survey

ZAHEED AHMED BHUIYAN, SALEKUL ISLAM (SENIOR MEMBER, IEEE), MD. MOTAHARUL ISLAM (MEMBER, IEEE), A B M AHASAN ULLAH, FARHA NAZ AND MOHAMMAD SHAHRIAR RAHMAN (MEMBER, IEEE)

Department of Computer Science and Engineering, United International University
United City, Madani Avenue, Badda, Dhaka 1212, Bangladesh.

Corresponding author: Mohammad Shahriar Rahman (e-mail: mshahriar@cse.uiu.ac.bd).

This work was supported by the United International University (UIU) Institute of Advanced Research (IAR) Research Grant Scheme under Grant IAR-2023-Pub-028.

ABSTRACT Software-Defined Networking (SDN) has revolutionized the networking landscape by offering programmable control and optimization of network resources. However, SDN architecture's inherent flexibility and centralized control expose it to new security risks. In this paper, we have presented a comprehensive study focused on the security implications associated with the control plane of SDN, which serves as a critical layer responsible for its network orchestration. We have addressed some pressing security concerns in SDN deployments by examining control plane vulnerabilities and explicit attacks. Through extensive analysis, we have investigated various control plane attacks. By meticulously exploring each attack vector, we have shed light on its mechanisms, potential impact and countermeasures. Furthermore, we have emphasized the interdependencies between the control plane, application plane, and data plane, highlighting how compromises in the control plane can propagate and impact the entire network infrastructure. Our research contributes to a deeper understanding of the specific vulnerabilities within SDN, focusing on the control plane as the primary target. By providing insights into the security landscape of SDN, network administrators, researchers, and security practitioners can develop proactive defense strategies and fortify the security posture of SDN deployments. We have underscored the importance of integrating robust security mechanisms to safeguard the control plane and maintain the overall security of SDN architectures. Our comprehensive analysis of control plane attacks in SDN elucidates the evolving security challenges posed by the programmability and centralization of network control. By addressing these vulnerabilities, we have tried to pave the way for future researchers to develop effective security solutions and ensure SDN networks' resilience and integrity.

INDEX TERMS Software-Defined Networking, SDN, Control Plane, Data Plane, Application Plane, SDN Controller, OpenFlow, SDN Attacks, DoS/DDoS Attacks

I. INTRODUCTION

Long before this design started to be utilized in data networks, the separation of the control and data plane was originally employed in the public switched telephone network to streamline the installation and management process.

In a proposed interface standard titled "Forwarding and Control Element Separation," released in 2004, the Internet Engineering Task Force (IETF) began exploring several options for separating the control and forwarding operations, which in short is known as ForCES [1]. The ForCES Working Group also suggested a related SoftRouter Architecture [2]. The Linux Netlink IP Services Protocol [3] and a path computation element (PCE)-based architecture are two other early standards from the IETF that sought to separate control from data.

The first instance of separating control and data plane designs using open-source software was the Ethane [4] project at Stanford University's computer sciences division. The simple switch architecture of Ethane led to the creation of OpenFlow. The initial OpenFlow API was developed in 2008 [5]. NOX, an OS for networks, was developed that same year [6].

In SDN frameworks, network control and forwarding tasks

are separated, allowing network control to be directly programmable and the supporting infrastructure insulated from the networking services and applications [7]. Some historical advancements in programmable networking and early SDN use cases are illustrated in Figure 1.

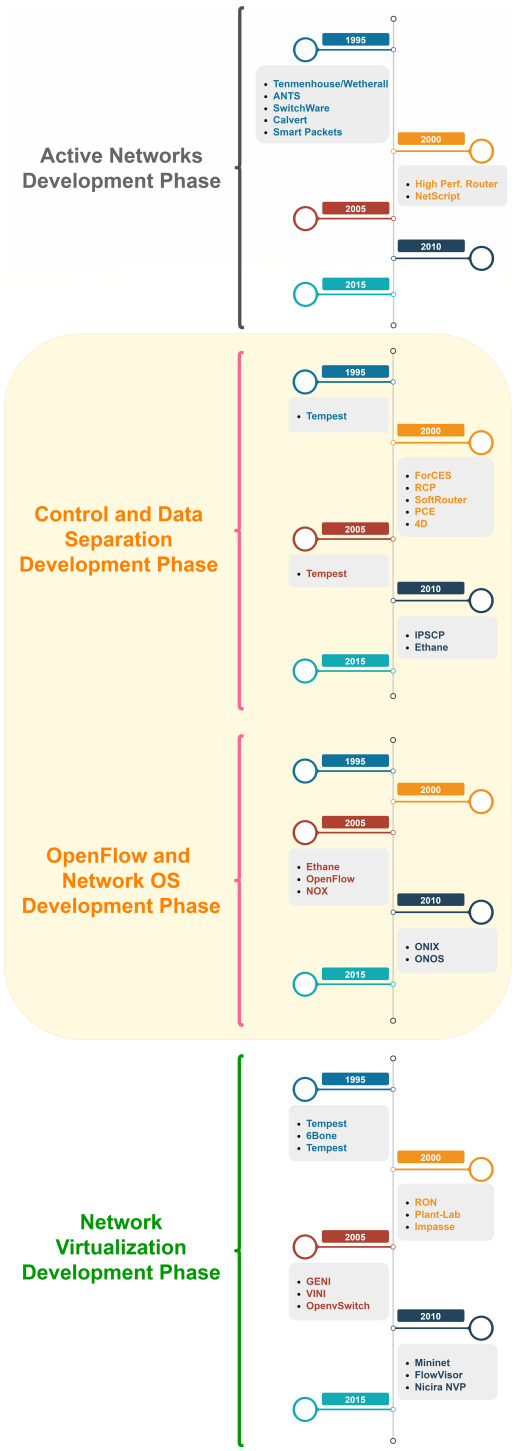

FIGURE 1. Chronology of major events during the evolution of SDN as well as programmable networking technologies [8]

SDN architecture is intended to be directly programmable, flexible, centrally controllable, programmatically adjustable, open standards-based, and vendor-neutral. Traditional approaches we usually follow to design and maintain networks change because of Software Defined Networking (SDN). Two factors distinguish SDN from traditional networking frameworks. An SDN splits the control plane from the data plane in the first place, with the control plane making decisions about how to manage traffic and the data plane forwarding that traffic in accordance with those decisions. An SDN also unifies the control plane, allowing a single software control program to manage numerous data-plane components. Through a well-defined Application Programming Interface (API), the SDN control plane directly manipulates the state of the network's data-plane components (such as routers, switches, and other middleboxes) [8]. Figure 2 illustrates how data would traditionally go via a networking infrastructure before the development of SDN, and Figure 3 illustrates how things have altered as a result of the development of SDN.

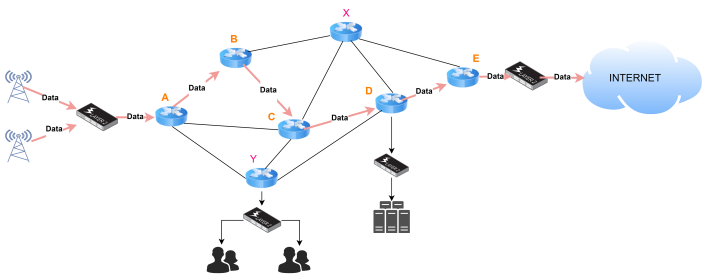

FIGURE 2. Data Travelling Scenario Before SDN

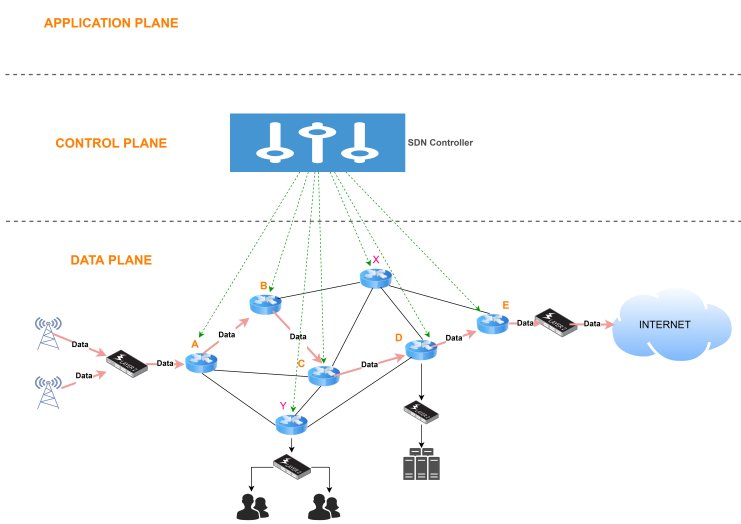

FIGURE 3. Data Travelling Scenario After SDN

The control layer of SDN and its component SDN controller are the most crucial elements of the SDN architecture, as is evident from the explanation above. As a result, even if it offers certain noteworthy benefits, it is also vulnerable to several threats. The SDN controller, the connection between two SDN controllers (in the case of multiple controllers in

the control layer), the northbound APIs at the northbound interface, which ensures communication with the controller, and the southbound APIs at the southbound interface, which ensures the communication with the controller can be considered the four main components of the attack surface [8]. Figure 5, Figure 6, and Figure 9 can help understand the idea.

This paper had made several contributions to the field of SDN security. The following bullet points outline the key contributions:

1) Attack classification and taxonomy: Our research classifies attacks against SDN control planes and organizes them based on different attack surfaces, including the Northbound Interface (NBI), the Southbound Interface (SBI), the SDN Controller, and the link between two SDN controllers (in multi-controller environment). This classification provides a structured understanding of the various attack vectors in SDN environments.
2) Taxonomical representation of findings: Our paper presents a taxonomical representation of the identified attacks, offering a systematic framework for analyzing and comprehending their characteristics. This taxonomical approach facilitates a clear and organized view of the attacks, aiding researchers and practitioners in understanding the relationships between different attack types.
3) Countermeasure taxonomies: In addition to attack classification, our research provides detailed countermeasure taxonomies aligned with the attack taxonomy. Our paper shows the corresponding countermeasures for each attack category to mitigate or prevent such attacks. These countermeasure taxonomies serve as practical guidance for implementing effective security measures in SDN environments.
4) Research gap analysis: The paper conducts a comprehensive research gap analysis, identifying limitations and research needs in SDN security. By highlighting these gaps, our research offers valuable insights for future researchers, enabling them to identify potential research directions and address the current shortcomings in the field.

These contributions enhance the understanding of attacks against SDN control planes by providing a structured classification, offering taxonomical representations, proposing countermeasure taxonomies, and identifying research gaps. The findings of this research paper contribute to the body of knowledge on SDN security and provide a foundation for future research and development in the field.

We have organized the paper in the following frameworks (Figure 4); after the abstract, Section I, Introduction, contains the historical evolution of SDN, motivation & contributions of the work, and general discussion. Section II describes the background of SDN and SDN working architecture. Section III describes the comparative study of the papers we have reviewed and a comparison table. Section IV describes the attack taxonomy where different attacks targeting different entry points SDN architecture are discussed. Section V describes the countermeasures for the attacks targeting different entry points SDN architecture. In Section VI, Distributed Denial of Service Attacks in the SDN environment has been discussed. Section VII describes some research gaps; Section VIII describes the future work parts. Sections IX describes the conclusion and is followed by references in Section X.

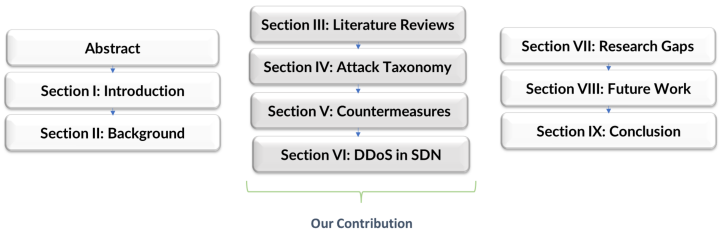

FIGURE 4. Organization of the Paper

II. BACKGROUND

A. SDN ARCHITECTURE

Some key features of SDN include the capability of configuring virtual networks, dynamic policy enforcement for networks, and a wider control for managing networks through a well-laid-out centralized console. The overall operational cost is also significantly lower than in conventional ways. It isolates the control logic from the network devices (switches and routers), aiming at substituting the conventional networks. The centralized control plane puts an extra burden on administrators to ensure overall network security and usual functionality. Compromised network objects can be a source to repossess delicate information regarding network structure and users. That information can later be used for unauthorized activities, such as bringing the network down.

The architecture of SDN is a layered approach, as shown in Figure 5. A detailed diagram is also displayed in Figure 6. SDN has three layers – i) (Network) Application layer, ii) Control layer, and iii) Infrastructure or data forwarding layer [9]. The application and control layers communicate between them using the Northbound API. The control layer and the data forwarding layer communicate using the Southbound API. OpenFlow protocol is the most common form of Southbound API in use. Different layers of SDN, along with their various components and functions, are briefly stated below –

1) Application Plane

In SDN (Software-Defined Networking) architecture, the Application Plane refers to the top layer of the SDN stack. It is responsible for hosting and executing network applications, management systems, and control applications that define network policies and behaviour. The Application Plane interacts with the Control Plane and the Infrastructure Plane to configure, monitor, and control the network.

The Application Plane is where network administrators and developers deploy and manage applications that utilize

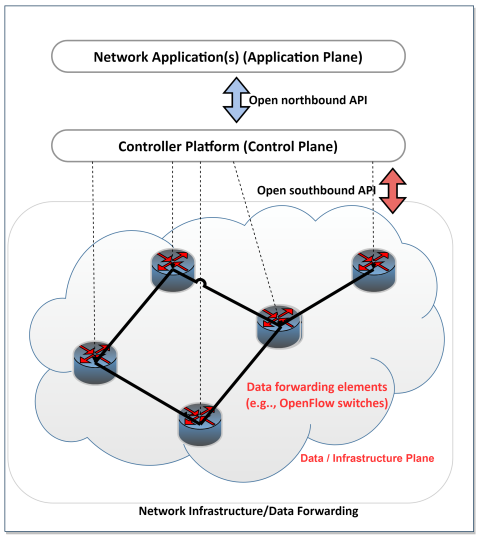

FIGURE 5. SDN Architecture (Generic)

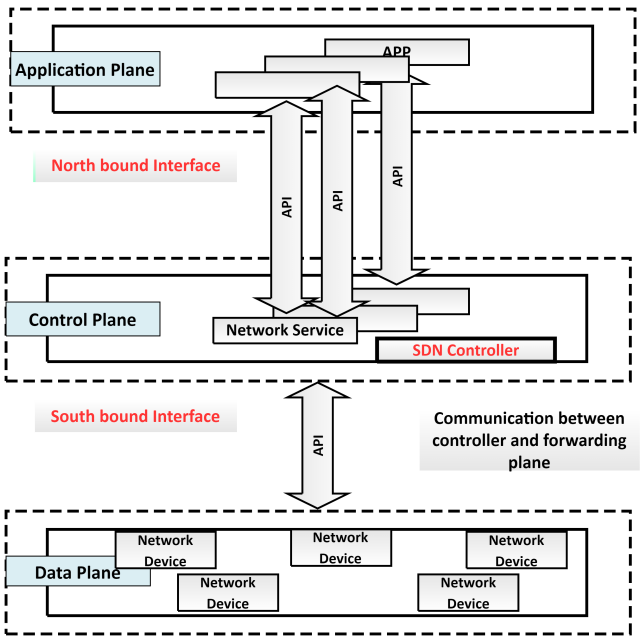

FIGURE 6. SDN Architecture (Comprehensive)

the programmability and flexibility of the SDN infrastructure. It allows for developing innovative network services and applications that dynamically control network behaviour based on specific requirements.

Key characteristics and functionalities of the Application Plane in SDN architecture include:

1) Application Development: The Application Plane provides a platform for developing and deploying network applications. It offers APIs (Application Programming Interfaces) [10] and software development kits (SDKs) that enable developers to create applications that control and manage the network.
2) Network Policy and Control: Applications in the Application Plane define network policies and rules for traffic forwarding, security, Quality of Service (QoS), and other network functions. These policies are implemented by communicating with the Control Plane to configure network devices accordingly.
3) Network Monitoring and Analytics: The Application Plane collects network data, monitors network performance, and performs analytics to gain insights into network behaviour and make informed decisions for network optimization and troubleshooting.
4) Service Orchestration: Applications in the Application Plane can orchestrate network services and resources to meet specific application requirements. This includes dynamic provisioning, scaling, and coordination of network functions and services.
5) Integration with Management Systems: The Application Plane interfaces with management systems, such as network management systems (NMS) or orchestration platforms, to provide a unified management and control framework for the network.

The Application Plane in SDN architecture plays a crucial role in enabling the deployment, management, and control of network applications and services. It provides a programmable interface for developers and administrators to define network behaviour, implement network policies, and leverage the flexibility of the underlying SDN infrastructure [9].

2) Control Plane

In SDN architecture, the control plane refers to the component responsible for managing and controlling the network. It is one of the three main components of SDN, alongside the data plane and the application plane.

The control plane is responsible for making decisions and implementing network policies that govern how data packets are forwarded within the network. It centralizes network intelligence and allows for programmability and flexibility in managing network operations. The control plane abstracts the network hardware and provides a logical network view, enabling network administrators to define and enforce network policies through software-based controllers.

The control plane in SDN architecture typically consists of one or more controllers. These controllers act as the brains of the network, overseeing the network operations and managing network devices such as switches and routers. They communicate with the data plane, which consists of network devices, through standardized protocols like OpenFlow [5], [7], Netconf [11].

The control plane performs several essential functions [12], including:

1) Network topology discovery: The control plane discovers the network topology by collecting information about connected network devices and their intercon-

nections. It maintains a network topology view, allowing for efficient routing and forwarding decisions.

2) Flow control and forwarding: The control plane defines and manages flow rules that dictate how data packets are forwarded within the network. It determines the optimal paths for packet routing and controls traffic flows based on defined policies and network conditions.
3) Network policy enforcement: The control plane enforces network policies by configuring and managing the behaviour of network devices. It ensures that traffic is classified, prioritized, and treated according to specified policies, such as Quality of Service (QoS) requirements or security measures.
4) Network orchestration and management: The control plane provides a centralized management interface for configuring and monitoring the network. It enables network administrators to provision network resources, set up virtual networks, and monitor network performance.

By separating the control plane from the data plane, SDN architecture offers advantages such as centralized management, programmability, and agility. It enables network administrators to dynamically adapt the network behaviour, automate network operations, and efficiently respond to changing requirements.

The control plane in SDN architecture plays a critical role in managing and controlling network operations, facilitating the software-defined nature of the network, and enabling efficient network management and automation [6], [13], [14].

3) Infrastructure or Data Forwarding Plane

In SDN architecture, the infrastructure or data forwarding plane is one of the three key components, along with the control and application planes. The data forwarding plane, also known as the data plane or forwarding plane, is responsible for the actual forwarding and processing of network traffic within an SDN network. In this layer, there is a coexistence of both virtual switches like Open vSwitch [15], Indigo [16], Pica8 [17], Nettle [18], Pantou [19], XorPlus [20] and physical switches [21]–[23].

The infrastructure or data forwarding plane consists of network devices such as switches, routers, and access points responsible for receiving, processing, and forwarding data packets based on the instructions from the SDN controller. These devices form the physical or virtual network infrastructure over which data flows.

In traditional networking architectures, the control and data planes are tightly coupled within each network device. However, in SDN, the data forwarding plane is decoupled from the control plane, allowing for centralized control and programmability of network behaviour. This separation enables dynamic network management and flexible traffic handling in SDN networks.

In an SDN architecture, the controller communicates with the infrastructure or data forwarding plane through the southbound interface (SBI). The controller instructs the network devices to handle and forward traffic by installing flow rules or policies in their forwarding tables. These flow rules define the desired behaviour for specific packets or flows, such as routing, traffic prioritization, and security policies.

The infrastructure or data forwarding plane plays a critical role in SDN as it is responsible for executing the forwarding decisions made by the SDN controller. It processes incoming packets, matches them against the installed flow rules, and determines the appropriate action, such as forwarding the packets to the intended destination or applying specific treatments or modifications to the packets.

SDN architecture provides flexibility, programmability, and centralized control over the network by separating the control plane from the data forwarding plane. This enables network administrators to efficiently manage and control network behaviour, optimize traffic flow, and implement advanced network services and policies.

Overall, the infrastructure or data forwarding plane in SDN architecture encompasses the network devices responsible for forwarding and processing network traffic based on the instructions received from the centralized SDN controller [9].

B. WORKING STAGES OF SDN

The working stages of SDN are represented in Figure 7. These below-mentioned 10 stages are required to complete a single packet transfer from host A to host D. However, when the controller has the flow rules installed, a packet will follow only stage 1, 4, 7 and 10 to travel from host A to host D. When a first packet travels from host D to host A it will follow the stages in reverse order.

C. OPENFLOW

OpenFlow is a protocol that facilitates the implementation of Software-Defined Networking (SDN) architecture by defining the communication between the control plane and the data forwarding plane. It provides a standardized interface for controlling network devices in an SDN network, such as switches, routers, and access points [5], [7].

Here are some key aspects of OpenFlow:

1) **OpenFlow Protocol:** OpenFlow uses a well-defined and standardized protocol for communication between the SDN controller and the network devices. The protocol specifies message formats, message types, and procedures for exchanging information and instructions.
2) **Centralized Control:** In an SDN architecture utilizing OpenFlow, the control plane is centralized in the SDN controller. The controller acts as the brain of the network, making intelligent decisions about how network traffic should be handled based on the network's overall state and policies.

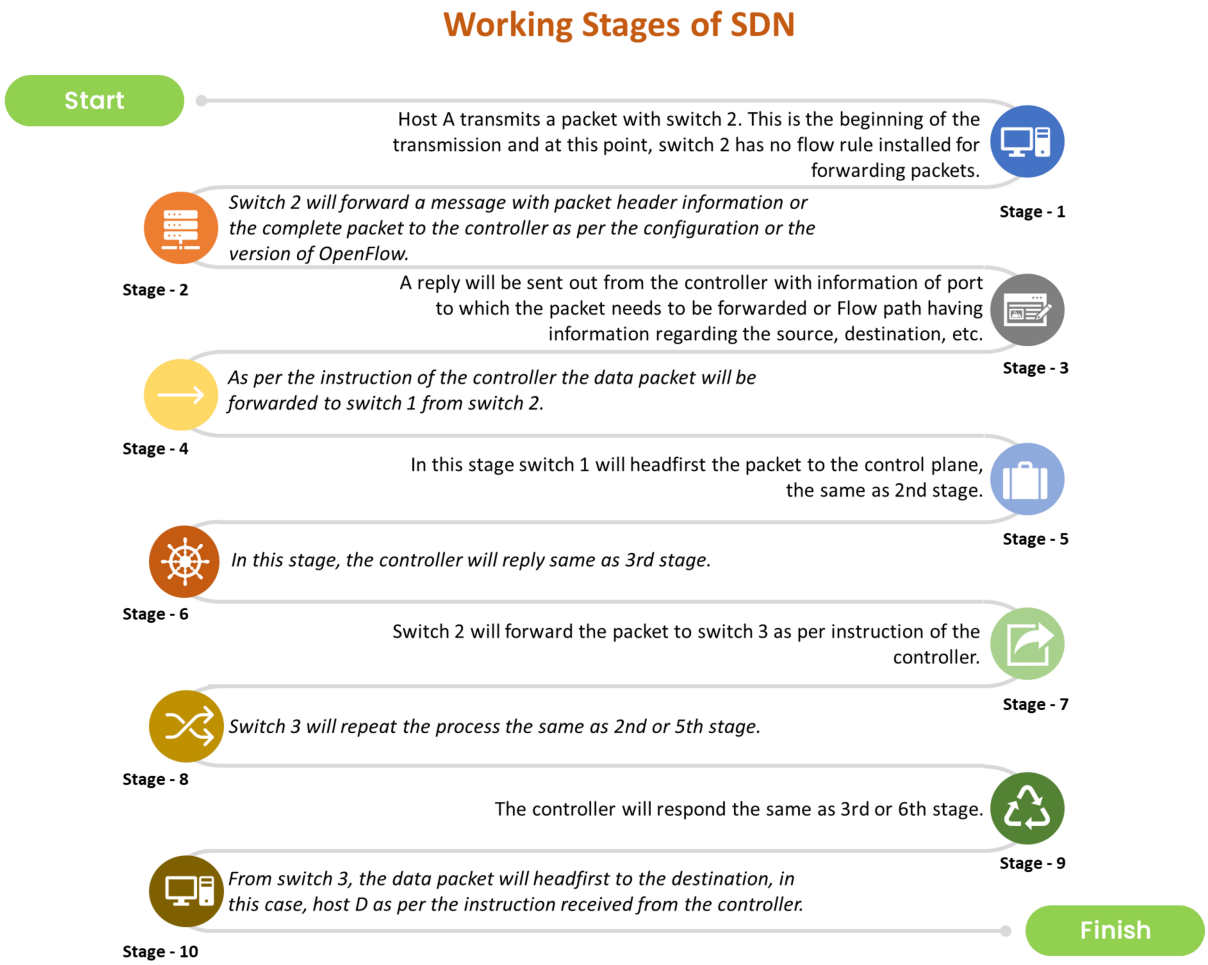

FIGURE 7. Working Stages of SDN

3) **Flow Tables and Flow Entries:** OpenFlow switches maintain flow tables, which are data structures that store flow entries. A flow entry consists of a set of match fields and corresponding actions. The match fields define the packet header attributes that the switch uses to match incoming packets, while the actions determine how the switch should process or forward the matched packets.
4) **Match Fields:** OpenFlow supports a range of match fields, including source and destination IP addresses, transport protocol (e.g., TCP, UDP), source and destination ports, VLAN tags, MPLS labels, and various other packet header fields. These match fields allow for fine-grained packet matching and control.
5) **Actions:** Each flow entry in the flow table has associated actions that specify how the switch should handle packets that match the entry. Actions can include forwarding packets to a specific port, modifying packet headers (e.g., rewriting MAC addresses, changing VLAN tags), applying QoS policies (e.g., setting packet priorities or bandwidth limitations), dropping packets, redirecting packets to the controller for further processing, or even invoking custom actions defined by network administrators.
6) **Flow Entry Installation and Modification:** The SDN controller uses the OpenFlow protocol to interact with OpenFlow switches through the southbound interface (SBI). It instructs switches to dynamically install, modify, or delete flow entries in their flow tables. This allows the controller to adapt network behavior based on changing network conditions, policies, or security requirements.
7) **Flow-Based Forwarding:** Once flow entries are installed in the switches' flow tables, the switches use hardware-based or software-based matching mechanisms to process incoming packets efficiently. They compare packet headers against the flow entries in their flow tables and execute the corresponding actions defined in the flow entries based on the match results.
8) **Programmability and Innovation:** OpenFlow's pro-

grammable nature enables network administrators, researchers, and developers to create custom network control applications, often referred to as network applications or network control programs. These applications interact with the SDN controller using the OpenFlow protocol, allowing for flexible and dynamic control over network behavior. This programmability fosters innovation, enabling the deployment of advanced network services, traffic engineering techniques, network slicing, network function virtualization (NFV), and other emerging networking paradigms.

9) **Standardization and Ecosystem:** OpenFlow has gained significant industry-wide adoption, becoming a widely accepted standard for SDN. A broad range of network equipment vendors and open-source software platforms supports it. The standardization of OpenFlow encourages interoperability, promotes the development of a diverse ecosystem, and fosters collaborative efforts in SDN research and deployment.

Overall, OpenFlow provides a standardized and extensible protocol for implementing SDN architecture. It enables centralized control, flow-based forwarding, dynamic network management, and programmability. It empowers network administrators and researchers to design, customize, and optimize network behavior based on evolving requirements and emerging technologies. In Figure 8, it displays a simplified version of the logical structure of an OpenFlow switch [24], [25].

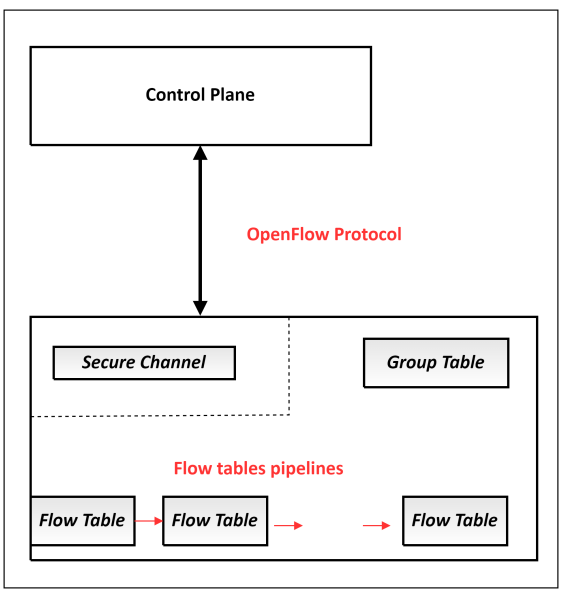

FIGURE 8. Simple Architecture of OpenFlow Switch

III. LITERATURE REVIEWS

In order to support our research framework, we reviewed related research papers in a particular way. The evolution of SDN and other programmable networks has first been researched from a historical perspective. Then, we have narrowed the scope of our research to include OpenFlow and SDN architecture. Finally, we have narrowed the scope of our research to the attacks that frequently target particular SDN architecture planes. Thanks to our review's final step, we have categorized the attacks related to the SDN control plane and illustrated our findings in the attack taxonomy and countermeasure sections by using the diagrams we have shown.

The conceptual history of programmable networks, encompassing active networks, early attempts to divide the control and data plane, and more recent work on OpenFlow and network operating systems, was documented and traced by N. Feamster et al. [8]. They emphasized fundamental ideas and the technological and practical pulls and pushed behind each breakthrough. They also discussed network virtualization, typical myths and misconceptions, and other technologies linked to SDN.

In their research, V. Thirupathi et al. [32] provided an overview of the advantages of employing SDN technology and the straightforward development of SDN and OpenFlow. Additionally, they summarized the need for OpenFlow in SDN design. To lessen traffic overhead to the controller for enabling NFV, Y. D. Lin et al. [33] developed an enhanced SDN architecture. They discovered that extending the OpenFlow specification to allow NFV modules in SDN is doable using their enhanced architecture. In their research, C. Janz et al. [34] looked at a few used scenarios for transport SDN (T-SDN), such as service bandwidth on demand, virtual transport network services, multi-layer control convergence, and resource optimization. Their analysis demonstrated that T-SDN might play a significant role in the utility framework's adoption of upcoming SDN technology. They also examined the multi-vendor T-SDN proof of concept. In their article, K. Raghunath et al. [35] evaluated existing defense strategies for SDN-enabled networks and tested those strategies on their own attack testbed. Additionally, they suggested possibly incorporating a defensive layer into the future SDN architecture's data plane. In their article, K. Cabaj et al. [36] examined the security implications of the SDN architectural components. To enhance its security capabilities, they suggested several improvements to SDN. They suggested a Distributed Frequent Sets Analyzer (DFSA) system, which employs SDN network properties, can be employed for effective and reliable detection of various network attacks.

S. Dong et al. [26] described a number of DDoS attacks against the SDN and cloud environments. They presented some unresolved issues with identifying and mitigating DDoS attacks, with a particular focus on SDN and cloud computing architecture. In their research, around 70 well-known DDoS detection and mitigation strategies in SDN networks were comprehensively reviewed by J. Singh et al. [27]. They divided these processes into four groups: information theory-based methods, machine learning-based

TABLE 1. Comparison Table for Related Works

Area	S. Dong et al. [26]	J. Singh et al. [27]	M. Arif et al. [28]	J. C. C. Chica et al. [29]	F. S. D. Silva et al. [30]	A. N. Alhaj et al. [31]	Our Work
Architecture of SDN	✓	✓	✓	✓	×	✓	✓
Attack Entry Points of Control Plane	✓	×	✓	✓	×	×	✓
Types of DDoS attacks in the Control Plane	✓	✓	×	×	✓	×	✓
Attacks Specifically Targeting the Controller	×	✓	×	✓	×	✓	✓
Attack Taxonomy	✓	✓	×	✓	✓	×	✓
Countermeasures	×	×	×	✓	✓	×	✓
Countermeasures for Controllers-based Attacks	×	×	×	×	×	×	✓
Countermeasures for NBI-based Attacks	×	×	×	×	×	×	✓
Countermeasures for SBI-based Attacks	×	×	×	×	×	×	✓
Countermeasures for Attacks between Controllers	×	×	×	×	×	×	✓
Research Gaps	×	✓	×	×	×	×	✓
Future Work	✓	✓	✓	×	✓	×	✓

methods, artificial neural network (ANN)-based methods, and other ad hoc methods. They also explained SDN's layered architecture thoroughly, outlining its advantages in preventing DDoS attacks and its weaknesses, allowing for developing new DDoS attacks instead of more traditional ones. In their article, M. Arif et al. [28] analyzed security threats that future SDN-based VANETs will have to deal with and looked at how SDNs could be helpful in developing new defenses against those threats. In their research, J. C. C. Chica et al. [29] described some security risks SDN faces and a list of attacks that prey on weaknesses, particularly incorrect configurations of SDN's fundamental elements. They also talked about the duality of SDN, which means that sometimes it is used specifically for security concerns, and other times there are concerns about security in the SDN architecture itself. They carried out a comprehensive survey to cover these issues. The Internet of Things (IoT) is becoming an emerging technology. Still, because of its size, it is challenging to implement security measures to protect against various attacks, particularly DDoS attacks. There have been past studies about using SDN to reduce DDoS assaults in IoT scenarios. F. S. D. Silva et al. [30] attempted to map out the existing solutions and their limitations and categorize them through a taxonomical representation in the hope that their survey may aid future researchers. Designing new solutions to mitigate these DDoS attacks is difficult as IoT technologies evolve quickly and become more complex. The research presented by S. M. Mousavi et al. [37] demonstrates how DDoS attacks particularly impact and deplete SDN controller resources in the control plane of SDN architecture. They also offered a method to identify such attacks based on the entropy fluctuation of the target IP address. Within the first 500 packets of the attack traffic, they could identify a DDoS attack.

In Table 1, we have shown some comparisons with related works.

IV. ATTACK TAXONOMY

In Figure 9, we designed an attack taxonomy showing different types of attacks that affect SDN control planes. We have grouped these attacks according to four attack entry points at the SDN control plane (North Bound Interface, Controller, South Bound Interface, and Link between two controllers). We have also designed other partial taxonomies showing respective countermeasures for the attacks in Figure 12, Figure 14, Figure 15, and Figure 16.

A. CONTROLLER BASED ATTACKS

Controller Based Attacks are described below:

1) Packet In Flooding (DoS/DDoS)

Due to the fact that these attacks are created by taking advantage of compromised controllers, corrupted switches, etc., they specifically target the southbound interface and the controller itself.

A new point of failure is added to the network due to centralizing the control plane. Using several controllers can minimize this, but controllers may still be vulnerable to denial-of-service (DoS) attacks without careful rule implementation. The control plane must handle some edge-case packets in present network devices. On the other hand, in OpenFlow, bad rule design might result in saturation levels of controller inquiries, which will impact all switches that depend on that controller.

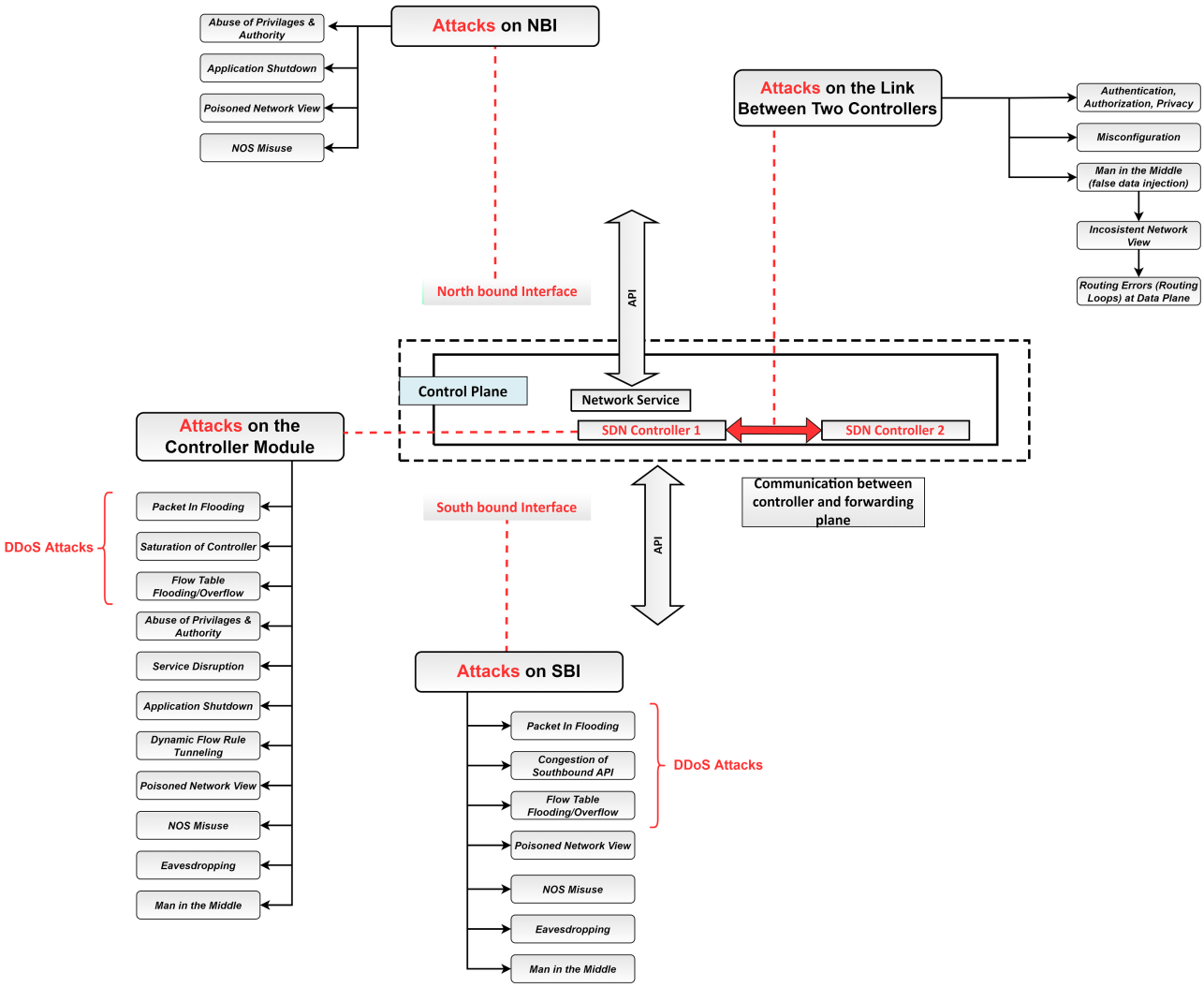

FIGURE 9. Attack Taxonomy

Most of these DoS threats affect networks that employ reactive policies. As soon as no traffic can produce random Packet-In events, networks focused on proactive rule placement do not have the same risk. These switches are still susceptible to a DoS brought on by a controller that makes too many flow adjustments. Developers of applications must take extra care to avoid situations that result in an abundance of Flow-Mod notifications. The OpenFlow 1.3 design recommends monitoring packets with a controller destination, but it also clarifies that this is not covered by definition. It does not guide rate-limiting signals to the controller and rule entries to the switches [38], [39].

2) Saturation of Controller (DoS/DDoS)

The Saturation of Controller attack is a Denial-of-Service (DoS) attack targeting the SDN controller in an SDN architecture. This attack aims to overwhelm the SDN controller's resources, causing it to become unresponsive and preventing it from managing network traffic effectively [40].

The SDN controller manages and orchestrates network traffic flows in an SDN architecture. The Saturation of Controller attack involves an attacker sending large traffic to the controller, typically via the Northbound API. This results in a high processing load on the controller, which can cause the resources of the controller to become overwhelmed, leading to a slowdown or even a complete shutdown of the controller.

This type of attack can be executed using various methods, such as network flooding or SYN flooding, where the attacker sends high traffic to the controller or exploits vulnerabilities in the controller software or firmware. The goal is to create congestion that ultimately affects the ability of the controller to manage network traffic, which can result in serious security concerns and compromise the integrity of the network [41].

To mitigate the Saturation of Controller attacks, network administrators can employ measures such as implementing rate-limiting mechanisms to restrict the number of requests

per second, monitoring network traffic patterns to detect anomalies, and deploying load-balancing techniques to distribute traffic across multiple controllers. Additionally, it is crucial to ensure that the controller software and firmware are up-to-date and apply access control policies to limit the number of users that can access the Northbound API [42].

Finally, it is important to have a response plan in place in the event of a Saturation of Controller attack, which includes procedures for isolating the affected controller and redirecting traffic to backup controllers or other components in the network. This can help minimize the attack's impact and ensure that the network remains available and responsive.

3) Flow Table Flooding/Overflow (DoS/DDoS)

These attacks target the southbound interface and the controller itself because they use faulty controllers, manipulated switches, etc. A scenario is shown in Figure 10

In software-defined networking (SDN), flow tables that route and analyze packets of flows are consumed by flow table overflow attacks, leaving no room for additional flows to implement flow rules and resulting in network denial of service (DoS). Such attacks pose severe security risks to SDN because they can be quickly launched by an enemy with hosts in the target network or have compromised those hosts [43], [44].

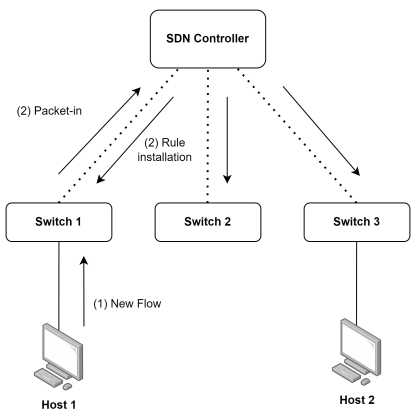

FIGURE 10. Flow Table Overflow Attack Scenario

4) Abuse of Privilages & Authority

These particular attacks, brought about by malicious SDN applications, impact the northbound interface (NBI) and the SDN controller itself. Researchers have illustrated the kind of attacks that rogue SDN applications can carry out.

To harm a NOS, malicious SDN applications (Sh14 [45], RH15a [46], RH [47]) take advantage of crucial system operations. Malicious SDN applications, for instance, can quickly crash SDN controllers, alter internal data structures, or create remote channels to access a C & C server and get shell commands, which are then performed on behalf of the NOS. Another case involves a rogue SDN application that downloads and runs any file with root access [48].

5) Service Disruption

These particular attacks, which have an impact on the SDN controller directly, come from a variety of domains.

By either faking northbound API communications or southbound communications to the network elements, the attacker would like to spawn new flows. If an attacker can effectively spoof flows from the authorized controller, they will have the power to alter how traffic moves through the SDN and may even be able to get around security-related regulations.

A DoS attack against the controller or some other technique to bring about the controller's inevitable failure might be attempted by an attacker. The attacker may utilize a resource-consumption attack to slow down the controller, making it react to Packet-In events very slowly and deliver Packet-Out messages gradually.

SDN controllers frequently use Linux-based operating systems. If the SDN controller uses a general operating system, the controller will also be vulnerable to such flaws. The default passwords and no security settings are frequently used when deploying controllers into production. The SDN engineers managed to "just barely" get it to function but were afraid of ruining it, so they didn't want to touch it. As a result, the system was left in production with a vulnerable configuration.

It would be problematic if an attacker could build their own controller and convince network components that data was coming from the "saboteur" controller. The attacker could then add items to the network devices' flow tables, preventing the SDN experts from seeing those flows from the standpoint of the commercial controller. In this scenario, the attacker would be in total control of the network [49], [50].

6) Application Shutdown

The northbound interface (NBI) is specifically targeted by these attacks caused by the compromised northbound protocol. Through NBI, the SDN controller also gets affected.

The SDN controllers employ a large number of northbound APIs. In addition to alternatives, northbound APIs could use Python, Java, C, REST, XML, and JSON. The controller would provide the attacker access over the SDN network if they could take advantage of the weak northbound API. A hacker might be able to develop their own SDN policies and take over the SDN ecosystem if the controller for the northbound API lacks any kind of protection.

A REST API [51] frequently has a predefined password that can be easily found. If an SDN implementation didn't alter this predefined password and a hacker could send packets to the controller's admin console, they might query the SDN

ecosystem's configuration and alter it to suit their needs [49], [52].

7) Dynamic Flow Rule Tunneling

These particular attacks, which are brought about by malicious SDN applications, impact the SDN controller itself. Researchers have illustrated the kind of attacks that rogue SDN applications are capable of carrying out.

A novel method is given in (Po12 [13], Po15 [53]) that enables attackers to go around flow controls that are present in an OpenFlow switch. The researchers showed that even when a current drop rule expressly forbids such a connection, an attacker can access a network host by introducing specially constructed malicious flow rules. Dynamic flow rule tunnelling is a method that uses the set and goto commands that are common in OpenFlow [48], [54].

8) Poisoned Network View

These particular attacks target the controller, the northbound, and the southbound interface because they use the link discovery service at the SDN control plane.

The control plane's provision of the link discovery service is essential to the efficient operation of network apps and services. The SDN controller's topology view can be tainted by an adversary by generating links across one or more infected devices.

Every Link Layer Discovery Protocol (LLDP) [55] packet that the controller receives is accepted, and its link data is used to update the controller's link information. This presents a security issue for the Link Discovery Service (LDS) [56]. More significantly, researchers discovered that the SDN controller's built-in approach does not safeguard the integrity or source of LLDP packets. Consequently, a hacker can simply alter the link data of the controller by injecting fake LLDP packets into the network or replaying real LLDP packets from one target switch to another [57], [58].

9) NOS Misuse

These particular attacks, which are brought about by malicious SDN applications, impact the southbound interface (SBI), the northbound interface (NBI), and the SDN controller itself. Researchers have illustrated the kind of attacks that rogue SDN applications are capable of carrying out.

For OpenDaylight [59], an SDN rootkit has been released (RH15b [60]) that provides the foundation for numerous enterprise solutions (SDC [61]). This SDN rootkit substantially modifies internal data structures to take over the components in charge of both programming the network and analyzing its state. Researchers have therefore shown that an attacker is capable of adding and hiding hostile flow rules, as well as removing valid flow rules, all without alerting the administrator. Additionally, a technique built on OpenFlow is given that permits remote communication between an attacker and the rootkit [62] component running inside the NOS. This is intriguing because the SDN design does not provide host communication between hosts running on the data plane and the control plane [48].

10) Eavesdropping

Due to the unencrypted control channel, these unique attacks target the southbound interface and the controller itself.

In SDN, eavesdropping assaults can occur within the data plane or through the communication lines connecting the controllers in the control plane and the forwarding devices in the data plane. Switches (also known as forwarding devices) and forwarding links are two places in the data plane where eavesdropping attacks occur. The malevolent eavesdroppers might monitor the data used for further attacks once they have corrupted and captured them. TCP networks frequently experience eavesdropping attacks [63].

11) Man in the Middle

These particular attacks target the southbound interface, the controller, and the connection between two controllers since they arise from unencrypted control channels, compromised SBI, and insecure data links.

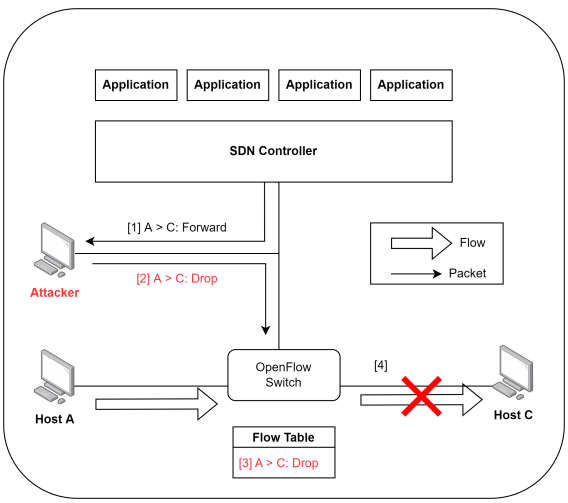

FIGURE 11. MITM attack scenario

Figure 11 depicts a MITM attack model wherein the ongoing OpenFlow messages on the control channel are actively manipulated by aggressively interfering with the interaction between the control plane and the data plane. When a flow rule tells the switch to convey a group of flows from host A to host C, (1) the controller delivers the rule, and (2) the attacker actively changes the action variable of the rule to "drop," In the end, the flow from host A to host C is dropped at the switch as a result of (3) the altered flow rule being installed on the switch [63] [64].

B. NORTH BOUND INTERFACE BASED ATTACKS

North Bound Interface Based Attacks are described below:

TABLE 2. Comparison Table for Attacks

Attacks	Controller	NBI	SBI	Link Between Controllers	Countermeasurers
Packet In Flooding	✓	×	✓	✓	Specialized programming in network devices
Saturation of Controller	✓	×	×	×	Rate Limiting, Load Balancing, Access Control, Firmware and Software Updates, Network Monitoring, Network Segmentation, Redundancy and Failover, etc.
Flow Table Flooding/Overflow	✓	×	✓	×	FlowVisor
Abuse of Privileges & Authority	✓	✓	×	×	Sandboxing
Service Disruption	✓	×	×	×	OS Hardening, Access Control, Role-Based Access Control (RBAC) regulations, High-Availability (HA) controller structure
Application Shutdown	✓	✓	×	×	Out-of-Band (OOB) system, TLS, SSH, Authentication, Encryption, Flow Comparison Testing
Dynamic Flow Rule Tunneling	✓	×	×	×	Sandboxing
Poisoned Network View	✓	✓	✓	×	LLDP Packet Authentication, Switch Port Property Validation
NOS Misuse	✓	✓	✓	×	Sandboxing
Eavesdropping	✓	×	✓	×	Multipath Method, Flow Table Integrity Verification, Forwarding Device-level Encryption
Man in the Middle	✓	×	✓	✓	SSL/TLS
Congestion of Southbound API	×	×	✓	×	Rate Limiting, Load Balancing, Access Control, Firmware and Software Updates, Network Monitoring, Network Segmentation, etc.
Authentication, Authorization, Privacy	×	×	×	✓	DISCO, Load Balancing Technologies, HyperFlow, McNettle, etc.
Misconfiguration	×	×	×	✓	Training of the Workforce, Data-at-rest encryption, Separation of Admin Privileged Accounts, Regular Patching, Checklist for Security Precautions, etc.

1) Abuse of Privileges & Authority

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.4 Section IV].

2) Application Shutdown

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.6 Section IV].

3) Poisoned Network View

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.8 Section IV].

4) NOS Misuse

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.9 Section IV].

C. SOUTH BOUND INTERFACE BASED ATTACKS

South Bound Interface Based Attacks are described below:

1) Packet In Flooding

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.1 Section IV].

2) Congestion of Southbound API

The Congestion of a Southbound API attack is a denial-of-service (DoS) attack [65] targeting the southbound interface of an SDN architecture. The attack's goal is to overwhelm the southbound API with a large volume of traffic, causing network traffic to become congested and preventing legitimate traffic from being processed.

The Southbound API communicates with the physical network switches and controls their behaviour. The Congestion of a Southbound API attack involves an attacker flooding the Southbound API with a large number of requests, which the controller cannot handle, causing the Southbound API to become congested. This results in delays and an inability to process legitimate traffic, leading to degraded network performance or even network failure.

This type of attack can be executed using various methods,

such as network flooding or SYN flooding [66], where the attacker sends a high volume of traffic to the Southbound API or utilizes many fake requests to overwhelm the interface. The goal is to create congestion that ultimately affects the ability of the controller to manage network traffic, which can result in serious security concerns and compromise the integrity of the network.

To mitigate the Congestion of the Southbound API attacks, network administrators can employ measures such as limiting the number of requests per second to the Southbound API, monitoring network traffic patterns, and implementing rate-limiting mechanisms that can help identify and prevent the attack. Additionally, the use of firewalls, intrusion detection and prevention systems, and other security technologies can help protect the SDN infrastructure against this type of attack [67]–[69].

3) Flow Table Flooding/Overflow

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.3 Section IV].

4) Poisoned Network View

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.8 Section IV].

5) NOS Misuse

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.9 Section IV].

6) Eavesdropping

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.10 Section IV].

7) Man in the Middle

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [See A.11 Section IV].

D. ATTACKS ON THE LINK BETWEEN TWO CONTROLLERS

Attacks on the link between two controllers are described below:

1) Authentication, Authorization, Privacy

The control plane is specifically impacted by these assaults, particularly the connections between controllers in a multi-controller system. The absence of authorization, improper authentication, and unencrypted communication channels contribute to such attacks.

SDN was first developed as a single controller architecture, which lacks scalability and dependability, to reduce the possibility of a single source of failure in the controller. As a result, the distributed control (controller clusters [70]) approach has been suggested, in which each controller instance serves as the master of a few switches, and various controllers can interact with one another to govern the entire network cooperatively. To the data forwarding layer, meanwhile, the presence of many hardware controllers operating the network rather than a single one should be invisible, which calls for the controllers to be seen as a unified controller for the overall network.

In this case, a networking application that covers numerous control areas will have to contend with several security issues, including those relating to network information transfer privacy, authentication, and authorization. The continuous switchover of the master controller and the coexistence of numerous controllers in a single network domain can also result in setting conflicts when several controllers work together in a dispersed manner. Inside the multi-controller architecture, an incorrect configuration is consequently also a covert security risk [71]–[73].

2) Misconfiguration

These attacks specifically impact the control plane, particularly the connections between controllers in a multi-controller system. These attacks are brought on by a system's improper design.

When a design setting or misconfiguration leaves an application module open to attack, this is known as a security misconfiguration vulnerability. Application subsystems or component misconfiguration exploits are configuration flaws that could exist. For example, the various attack vectors could make use of misconfiguration flaws:

1) **Buffer Overflow:** In an SDN environment, a buffer overflow attack due to misconfiguration can occur when the controller or network device is misconfigured to allow an attacker to send a large amount of data to a buffer that is not large enough to handle it. This can cause the buffer to overflow, leading to system instability or even crashes. Attackers can exploit this vulnerability by sending malicious packets with specially crafted payloads designed to trigger the buffer overflow. A buffer overflow attack can be hazardous, enabling an attacker to execute arbitrary code on the system, potentially taking control of the network or stealing sensitive data. This attack can be difficult to detect, especially if the attacker avoids triggering system crashes.

   To mitigate this attack, SDN administrators should ensure that all network devices and controllers are properly configured with sufficient buffer sizes. It is also essential to keep all software up-to-date with the latest security patches and to use intrusion detection and prevention systems to detect and prevent malicious traffic. Regular security audits and vulnerability

assessments can also help identify and mitigate buffer overflow vulnerabilities before they can be exploited [74], [75].

2) **Code Injection:** Code injection attacks due to misconfigurations can occur when the controller software is not configured securely, which allows attackers to insert malicious code into the SDN controller. This can occur due to weak passwords, unsecured interfaces, or unpatched software. Once the attacker has injected malicious code, they can potentially take control of the controller and launch other attacks, such as Denial of Service or stealing sensitive data. The attacker could also modify the controller's programming to disrupt network traffic, leak confidential data, or cause permanent damage to the network. For instance, the attacker could inject a code that bypasses access control to allow unauthenticated users access to sensitive network resources or modify the controller's decision-making process, which can lead to the network's failure or unauthorized access to the system. To perform the code injection attack, the attacker may take advantage of software vulnerabilities, including SQL injection [76] or Cross-Site Scripting (XSS) [77], to inject malicious code into the controller's software.

   To mitigate this type of attack, the controller software and all its dependencies must be updated with the latest patches and updates to prevent vulnerabilities. The controller should be configured to prevent unauthorized access and use strong authentication mechanisms to reduce the risk of weak passwords. Additionally, the controller should be isolated from the rest of the network using access control lists or firewalls to prevent unauthorized access to the controller interfaces. The controller should also have a secure boot process and be validated before deployment to ensure no malicious code has been injected. Lastly, the controller's software should be reviewed periodically to detect vulnerabilities or security weaknesses [78]–[80].

3) **Credential stuffing/Brute Force:** A credential stuffing/brute force attack is an attack in which an attacker attempts to gain unauthorized access to a network by repeatedly trying different username and password combinations until they find the correct one. This attack can be performed due to misconfiguration of the SDN network, such as weak or default passwords on network devices or software components used in the SDN environment. The attacker can use automated tools to test large lists of possible username and password combinations in a short period of time. If successful, the attacker can access the SDN network, allowing them to control the network and launch further attacks. This can lead to network downtime, data breaches, and other security risks.

   To mitigate this attack, SDN networks should use strong, complex passwords for all network devices and software components. Multi-factor authentication should also be used wherever possible to increase the security of the SDN network. Additionally, systems should be monitored for unusual login attempts, and login attempts should be rate-limited to prevent brute-force attacks [81]–[83].

4) **Command Injection:** In an SDN environment, a command injection attack due to misconfiguration can occur when an attacker injects malicious commands into the network device. This attack takes advantage of vulnerabilities in the input validation process and allows attackers to execute unauthorized commands on the device. To launch this attack, the attacker typically exploits the network device's web interface, which may have weak authentication mechanisms, default passwords, or other misconfigurations that make it vulnerable. The attacker can use these weaknesses to bypass authentication or use brute force to guess the login credentials. Once authenticated, the attacker can enter malicious commands that the device will execute, giving the attacker complete control over the network.

   This attack can have severe consequences, such as the ability to take down the entire network or compromise sensitive information. To prevent command injection attacks, following security best practices such as using strong passwords, disabling unnecessary services, and keeping the network device firmware up to date is crucial. Additionally, input validation should be implemented to ensure all user input is properly sanitized to prevent malicious commands from being executed. Network devices should also be monitored for any suspicious activity, and traffic analysis tools can be used to detect and block any unauthorized traffic [84]–[87].

5) **Cross-site Scripting (XSS):** Cross-site scripting (XSS) is an attack where an attacker injects malicious code into a vulnerable web application executed by the victim's browser. This can occur in web-based interfaces, such as the controller GUI, used to manage the SDN network in an SDN environment. In this attack, the attacker injects JavaScript code into a web page or form field, which then executes in the context of the victim's browser when the page is loaded, or the form is submitted. The injected code can steal cookies or session tokens, redirect the victim to a malicious site, or modify the page's content. This attack can occur due to misconfiguration of the web server, application server, or SDN controller software. An attacker can upload and execute malicious code if the web server or application server is misconfigured. An attacker can inject code into the web-based management interface if the SDN controller software is not adequately secured.

   Mitigation techniques for this attack include input val-

idation and output encoding, which can prevent malicious code injection. Using secure coding practices and frameworks can also help prevent XSS attacks. Additionally, enforcing proper access control policies and limiting the use of privileged accounts can reduce the impact of an XSS attack. Regular security audits and penetration testing can also help identify and remediate vulnerabilities [77], [88], [89].

6) **Forceful Browsing:** Forceful browsing, also known as a directory traversal or path traversal, is an attack in which an attacker tries to access files and directories outside the intended directory or file system. Misconfigurations can cause this attack in the SDN application or web server, which allows the attacker to manipulate the URL to access restricted files and directories. In SDN, forceful browsing attacks can compromise the network's security by allowing attackers to access sensitive information or configurations stored in the network devices. An attacker can use this information to gain unauthorized access to the network, modify network configurations, or cause a denial of service.

   To prevent forceful browsing attacks in SDN, several countermeasures can be taken. One of the most critical countermeasures is to sanitize and validate all user input to prevent the manipulation of URLs. The SDN application or web server should check for invalid characters, including "../" or "%2e%2e%2f", which can be used to escape from the intended directory. Another countermeasure is configuring access controls and permissions on the directories and files to prevent unauthorized access. The SDN application should enforce strict access controls on sensitive directories and files, allowing only authorized users to access them. Access controls should be reviewed periodically to ensure they are still effective. Additionally, web application firewalls can prevent forceful browsing attacks by filtering out malicious traffic before it reaches the SDN application or web server. These firewalls can detect and block attempts to escape from the intended directory or manipulate the URL. Regular security audits and penetration testing can also help to identify vulnerabilities that forceful browsing attacks could exploit. These tests can identify misconfigurations in the SDN application or web server and provide guidance on improving security and preventing future attacks [90], [91].

3) Man in the Middle

This attack has been discussed above in the subsection 'Controller Based Attacks' under the 'Attack Taxonomy' section [A.11 Section IV].

V. COUNTERMEASURES

A. FOR CONTROLLER-BASED ATTACKS

A taxonomy is shown in Figure 12, where the countermeasures are mentioned according to the SDN controller-based attacks.

1) Countermeasure for Packet In Flooding

Modern network devices feature specialized programming to address the recognized weaknesses of the protocols they support. For instance, the majority of modern enterprise Ethernet switches contain code that prevents DHCP snooping, broadcast/multicast rate limiting, and port-level MAC address restrictions. The controller in OpenFlow networks must offer all of these fundamental safeguards. Consequently, the developers of the OpenFlow apps, who might not be aware of the existence of these assaults, are left with the responsibility of putting in place complicated security safeguards [38].

2) Countermeasure for Saturation of Controller

Network administrators can employ various techniques to mitigate the Saturation of Controller attacks to ensure that the SDN controller can handle large traffic volumes and prevent an attacker from overwhelming the controller's resources. Some of the mitigation techniques include:

1) **Rate Limiting:** Implementing rate limiting mechanisms to restrict the number of requests per second sent to the SDN controller can help control the traffic flow and reduce the controller's load. By limiting the rate of incoming traffic, the controller can process incoming requests more efficiently and prevent a single traffic source from overwhelming the controller [92]–[94].

2) **Load Balancing:** Load balancing can help to distribute traffic across multiple controllers and prevent a single controller from becoming overwhelmed. This can help ensure the network remains available and responsive, even under heavy traffic conditions [95].

3) **Access Control:** Network administrators can ensure that only authorized users and devices can access the SDN controller by implementing access control policies. This can help to prevent attackers from sending traffic to the controller, which could cause congestion and disrupt network traffic [96], [97].

4) **Firmware and Software Updates:** Regularly updating the firmware and software used in the SDN infrastructure can help to ensure that known vulnerabilities and exploits are addressed. This can help to prevent attackers from exploiting weaknesses in the infrastructure to launch the Saturation of Controller attack [98].

5) **Network Monitoring:** Implementing network monitoring tools that provide visibility into network traffic patterns and detect anomalies that may indicate an attack is in progress. This can include real-time traffic analysis, alerting, and reporting tools [99].

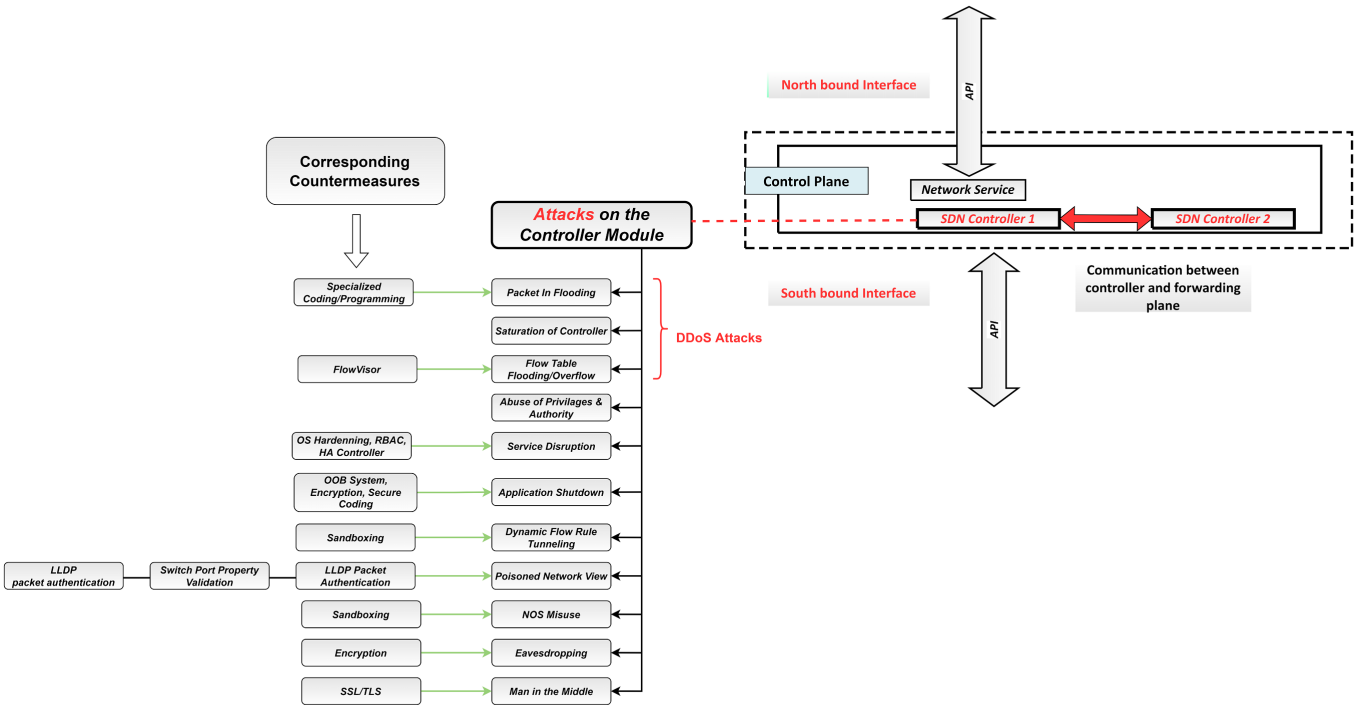

FIGURE 12. A taxonomy for the countermeasures related to the controller-based attacks

6) **Network Segmentation:** Segmenting the network into smaller subnetworks can help to reduce the impact of the Saturation of Controller attack. This can be achieved by deploying firewalls or other security technologies that monitor network traffic and detect and block malicious traffic that attempts to overload the SDN controller [100], [101].

7) **Redundancy and Failover:** Implementing redundancy and failover mechanisms can help ensure the network remains available and responsive, even if one or more network components fail. This can involve deploying multiple controllers, backup systems, or other components to provide redundancy and failover capabilities [100], [102].

By implementing these best practices, network administrators can help to mitigate the risk of the Saturation of Controller attacks and ensure that the SDN infrastructure remains secure and available.

3) Countermeasure for Flow Table Overflow

Network administrators can distinguish between distinct network packets using FlowVisor by looking at the header sections of the packets. Between switches and controllers, FlowVisor serves as an intermediary. It receives rules from controllers and revises them such that the resultant rules only apply to the areas of the network that a particular controller is permitted to operate. For instance, a controller might be given access to the network segment that carries all traffic to and from a company's web servers. In response to a DoS assault, this controller might establish a rule to stop all UDP communication. All UDP traffic to and from the web servers will be dropped when FlowVisor gets this rule, leaving the rest of the network untouched [71].

4) Countermeasure for Abuse of Privileges and Authority (Sandboxing)

There are now two suitable sandbox systems for SDN controllers. The first manages access to system calls while running SDN applications in different processes [Sh14]. The other system [RH15a, RH, SDa] uses Java security capabilities to lock SDN applications inside Java sandboxes. Figure 13 depicts the fundamental protection mechanism, which is the same for both approaches.

The access to essential operations (such as system calls or delicate Java operations) is managed by a NOS, and each SDN application operates in its own isolated sandbox. The sandbox architecture, or which SDN application is permitted to carry out which crucial tasks, must be supplied by a network administrator. If he or she gives access to a vital operation, the associated SDN application can carry it out or refuse access [48].

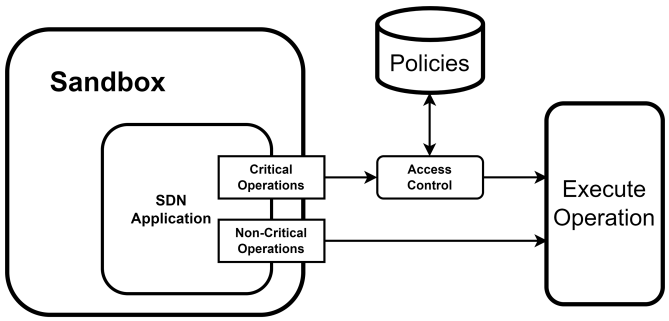

FIGURE 13. Sandboxing

This approach can also help mitigate the following attacks:

- Dynamic Flow Rule Tunneling (See A.7 Section IV)
- NOS Misuse (See A.9 Section IV)

5) Countermeasure for Service Disruption

Since the controller is a prime attack target, it must be strengthened. The host OS needs to be hardened to improve the defense capabilities of the controller and the networking devices. The same best practices apply to hardening Linux systems with a public-facing are applicable here. However, businesses should keep a tight eye on their controls for any unusual behaviour.

The SDN control system should not be open to unwanted access, according to organizations. SDN solutions ought to provide setting up secure, verified admin privileges to the controller. Controller admins might need to use Role-Based Access Control (RBAC) [103] regulations. Checking for illegal changes made by admins or attackers alike may be made possible through logging and audit trails.

A High-Availability (HA) controller structure is advantageous in the event of a DoS assault on the controller. SDNs that employ redundant controllers may lose one controller while still operating. The difficulty level for an attacker attempting to DoS every controller in the system would rise. Furthering the attacker's desire to avoid detection, that attack would also not be especially discreet [49].

6) Countermeasures for Application Shutdown

Another defense mechanism is utilizing an Out-of-Band (OOB) system [104] to regulate traffic. An OOB network can be built more easily and affordably in a data center than across a corporate WAN. An OOB system for northbound and southbound communications might be safer while managing controllers.

It would be recommended as the best approach to encrypting controller operations and northbound connections using TLS [105], SSH [106], or another technique. Authentication and encryption techniques should be used to protect communications from software programs and services that ask the controller for data or services.

All northbound applications that ask for SDN resources should follow secure coding standards. Secure programming techniques are helpful for the security of Internet web apps that are accessible to the public and apply to northbound SDN interfaces.

A few SDN systems can compare flows in network device tables to controller rules. This form of testing (similar to FlowChecker) of the network devices' flows versus the policy inside the policy could aid in locating differences that are the consequence of an attack [49].

7) Countermeasures for Dynamic Flow Rule Tunneling

Please see 'Sandboxing' under the subsection 'For Controller-based Attacks' [See A.4 Section V].

8) Countermeasures for Poisoned Network View

The authenticity of an LLDP packet can be breached during the link discovery process in OpenFlow networks, and infected hosts can get involved in the LLDP propagation path, according to a summary of the root causes of the Link Fabrication exploits. Researchers provided two methods to protect the Link Discovery process without needing manual labour to address those security lapses.

1) **LLDP packet authentication:** An attacker's initial security flaw is that the OpenFlow controller doesn't check the validity of LLDP packets. As long as the attacker can obtain LLDP packets from the linked switch, he or she can also undermine the authentication of the origin in contemporary OpenFlow controllers. Increasing the number of identifier TLVs (Type-Length-Values) in the LLDP packet is one way to address this issue. In particular, while receiving LLDP packets, we can concatenate a controller-signed TLV and verify the identity. The contents of the LLDP packet—specifically, the DPID (Datapath ID) and Port number—are used to calculate the signature TLV. The attacker has less ability to alter the LLDP packets in this situation. The drawback of this strategy is that it cannot counter an LLDP relay/tunnelling attack called Link Fabrication [58].

2) **Switch port property validation:** The fact that no hosts can take part in the LLDP propagation is another security fundamental of the OpenFlow link discovery process. Checking whether any hosts are present within the LLDP propagation is one method to reduce the relay-based Link Fabrication. For example, we might add some additional logic to track the traffic from each switch port to determine which device has been connected to the port. When host-generated traffic (like DNS) is detected by OpenFlow controllers coming from a particular switch port, we set that port's Device Type to HOST. If not, when LLDP packets are obtained from such switch ports, we designate those switch ports as SWITCH [58].

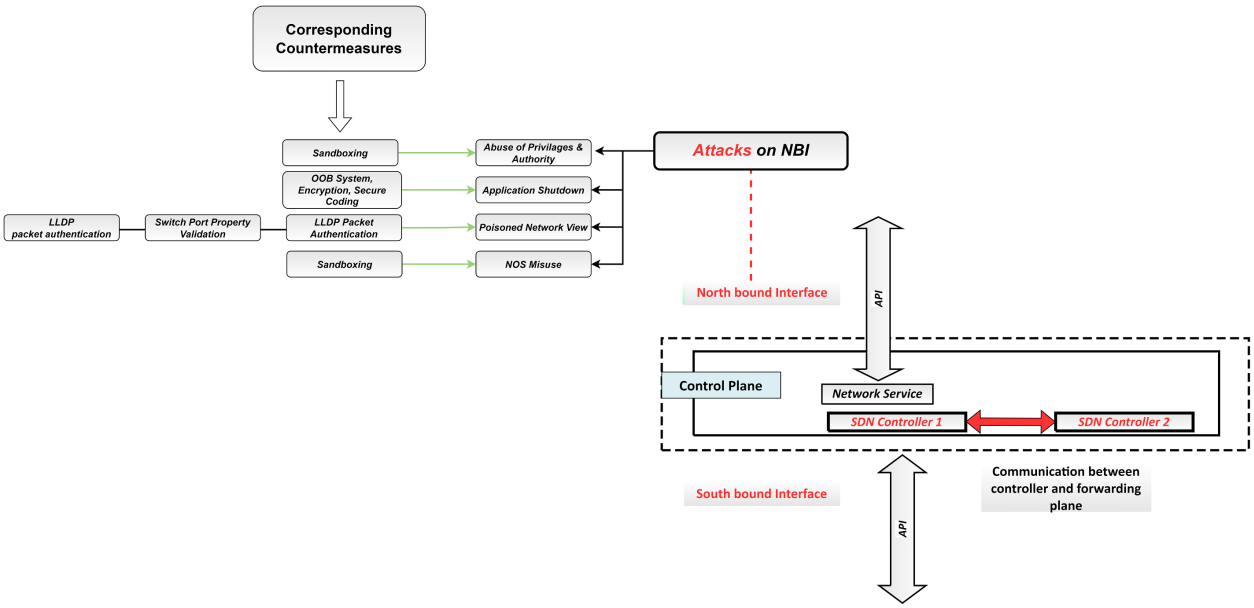

FIGURE 14. A taxonomy for the countermeasures related to the NBI-based attacks

3) **LLDP can only broadcast** on switch internal link ports and ports attached to the OpenFlow controller. Hence those two groups are essentially exclusive in OpenFlow networks. This approach makes the premise that the compromised computer is not a switch and will therefore continue to produce host-generated traffic (e.g., ARP, DNS). This presumption is sensible, and it holds in the majority of situations in reality. While a strong attack could theoretically stop all host-generated traffic in infected devices or virtual machines, it could also render the machine partially inoperable, at least for some standard networking tasks, and such a non-functional irregularity could be immediately noticeable by the regular machine user, thereby exposing the attacker's presence [58].

9) Countermeasures for NOS Misuse

Please see 'Sandboxing' under the subsection 'For Controller-based Attacks' [See A.4 Section V].

10) Countermeasures for Eavesdropping

Although eavesdropping attempts can be prevented by encryption, eavesdropping detection techniques are not yet widely available. Anti-eavesdropping measures in SDN cover the same fundamental steps as in traditional networks, from prevention to detection and to lessen the severity. The phases include a multipath method, flow table integrity verification, and forwarding device-level encryption [71].

11) Countermeasures for Man in the Middle

Due to OpenFlow's lack of implementation of the control message integrity verification technique, active flow manipulation throughout the man-in-the-middle attack was permitted. If SSL/TLS security is enabled, this approach is not required. However, the SSL/TLS technology that is now available is insufficient to secure big SDN networks [63].

B. FOR NBI-BASED ATTACKS

A taxonomy is shown in Figure 14, where the countermeasures are mentioned according to the SDN Northbound Interface (NBI) based attacks.

1) Countermeasure for Abuse of Privilages Authority

Please see 'Sandboxing' under the subsection 'For Controller-based Attacks' [A.4 Section V].

2) Countermeasure for Application Shutdown

Please see 'Countermeasures for Application Shutdown' under the subsection 'For Controller-based Attacks' [A.6 Section V].

3) Countermeasure for Poisoned Network View

Please see 'Countermeasures for Poisoned Network View' under the subsection 'For Controller-based Attacks' [A.8 Section V].

4) Countermeasures for NOS Misuse

Please see 'Sandboxing' under the subsection 'For Controller-based Attacks' [A.4 Section V].

C. FOR SBI-BASED ATTACKS

A taxonomy is shown in Figure 15, where the countermeasures are mentioned according to the SDN Southbound Interface (SBI) based attacks.

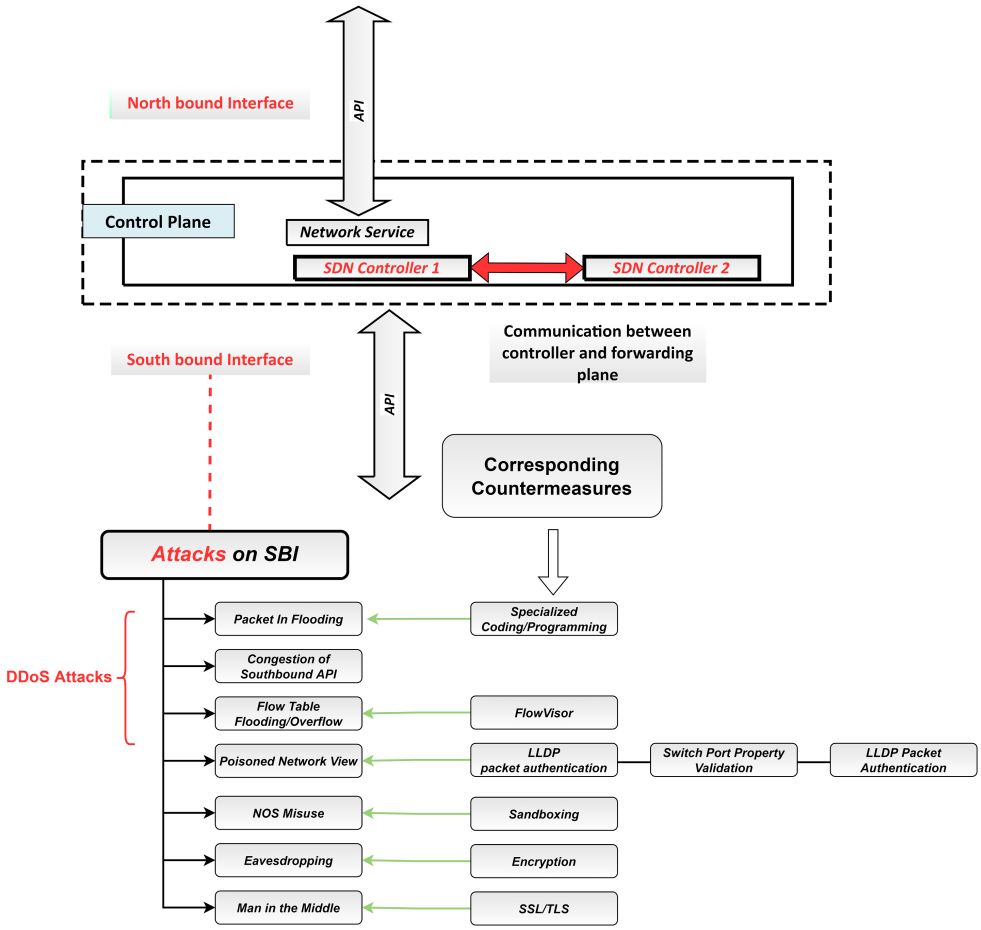

FIGURE 15. A taxonomy for the countermeasures related to the SBI-based attacks

1) Countermeasure for Packet In Flooding

Please see 'Countermeasures for Packet In Flooding' under the subsection 'For Controller-based Attacks' [A.1 Section V].

2) Countermeasure for Congestion of Southbound API

The Congestion of a Southbound API attack is a denial-of-service (DoS) attack targeting the Southbound API of an SDN architecture. To mitigate this type of attack, network administrators can implement various techniques and best practices, including:

1) **Rate Limiting:** To limit the amount of traffic allowed to the Southbound API, network administrators can deploy rate-limiting mechanisms that restrict the number of requests per second. This can help identify and prevent the Congestion of the Southbound API attack by controlling the traffic flow and reducing the overall load on the controller [92], 60, 61].
2) **Access Control:** Network administrators can ensure that only authorized users and devices can access the Southbound API by implementing access control policies. This helps to prevent attackers from flooding the interface with requests that could cause congestion and disrupt network traffic [96], [97].
3) **Segmentation:** Segmenting the network into smaller subnetworks can help to reduce the impact of the Congestion of Southbound API attacks. This can be achieved by deploying firewalls or other security technologies that monitor network traffic and detect and block malicious traffic that attempts to overload the Southbound API [100], [101].
4) **Network Monitoring:** To detect and respond to the Congestion of Southbound API attacks, network administrators can implement network monitoring tools that provide visibility into network traffic patterns and detect anomalies that may indicate an attack is in progress. This can include real-time traffic analysis,

alerting, and reporting tools [99].

5) **Load Balancing:** Load balancing can be used to distribute traffic across multiple controllers to prevent any one controller from becoming overloaded. This can help ensure the network remains available and responsive, even under heavy traffic conditions.

6) **Firmware and Software Updates:** Regularly updating the firmware and software used in the SDN infrastructure can help to ensure that known vulnerabilities and exploits are addressed. This can help to prevent attackers from exploiting weaknesses in the infrastructure to launch the Congestion of Southbound API attacks [98].

By implementing these best practices, network administrators can help to mitigate the risk of the Congestion of Southbound API attacks and ensure that the SDN infrastructure remains secure and available.

3) Countermeasures for Flow Table Flooding/Overflow

Please see 'Countermeasures for Flow Table Overflow' under the subsection 'For Controller-based Attacks' [A.3 Section V].

4) Countermeasure for Poisoned Network View

Please see 'Countermeasures for Poisoned Network View' under the subsection 'For Controller-based Attacks' [A.8 Section V].

5) Countermeasures for NOS Misuse

Please see 'Sandboxing' under the subsection 'For Controller-based Attacks' [A.4 Section V].

6) Countermeasure for Eavesdropping

Please see 'Countermeasures for Eavesdropping' under the subsection 'For Controller-based Attacks' [A.10 Section V].

7) Countermeasures for Man in the Middle

Please see 'Countermeasures for Man in the Middle' under the subsection 'For Controller-based Attacks' [A.11 Section V].

D. FOR ATTACKS BETWEEN TWO CONTROLLERS

A taxonomy is shown in Figure 16, where the countermeasures are mentioned according to the SDN attacks that happen between two controllers.

1) Countermeasures for Authentication, Authorization, Privacy Attacks

A specific protocol known as the Advanced Messaging Queuing Protocol [107] is used in DISCO's [108] implementation, which is built on Floodlight [109] and delivers control plane services for dispersed heterogeneous networks. An intra-domain control module and an inter-domain control module make up DISCO. The inter-domain control module keeps track of and controls the importance of data transferred across the domains so that flow pathways with various priorities can be determined and transmitted. The inter-domain control module can dynamically reroute or block traffic flow to combat attacks. The intra-domain control module, which consists of a message transceiver and several agents, is in charge of controlling communication between controllers. The message transceiver aims to detect nearby controllers and offer a control channel that limits controller interaction. The message transceiver module's communication channel allows agents to interchange data for the network. As we can see, the DICSO can adequately address the security risks that distributed controllers confront [71].

Other countermeasures (Controller/Controller Platform/Method) besides the one mentioned above are Load Balancing Technologies [110], HyperFlow [111], McNettle [112], and others.

2) Countermeasures for Misconfiguration

Several steps can be taken to prevent misconfiguration [113]:

1) Training as well as educating the workforce on current security developments, is one of the best ways to prevent security misconfiguration.

2) The data exfiltration files' security may be aided by using data-at-rest encryption techniques. For folders and files, we can also implement the proper access controls. These safeguards mitigate the susceptibility of the files and folders.

3) Systems can identify vulnerabilities automatically by running security scans. After making architectural improvements, conducting such scans regularly is an important step in reducing the net vulnerability.

4) Make sure admins have separate accounts for when they use their admin privileges compared to when they use the system normally.

5) To lessen the attack vectors, a regular patching schedule must be established, and updated software must be maintained.

6) Creating a checklist that includes the various security precautions you wish to take to be sure we've covered all the bases.

3) Countermeasures for Man in the Middle

Please see 'Countermeasures for Man in the Middle' under the subsection 'For Controller-based Attacks' [A.11 Section V].

VI. DISTRIBUTED DENIAL OF SERVICE ATTACKS IN SDN ENVIRONMENT:

We have introduced this separate part to talk about the attack scenario as well as some typical DoS/DDoS attacks on SDN

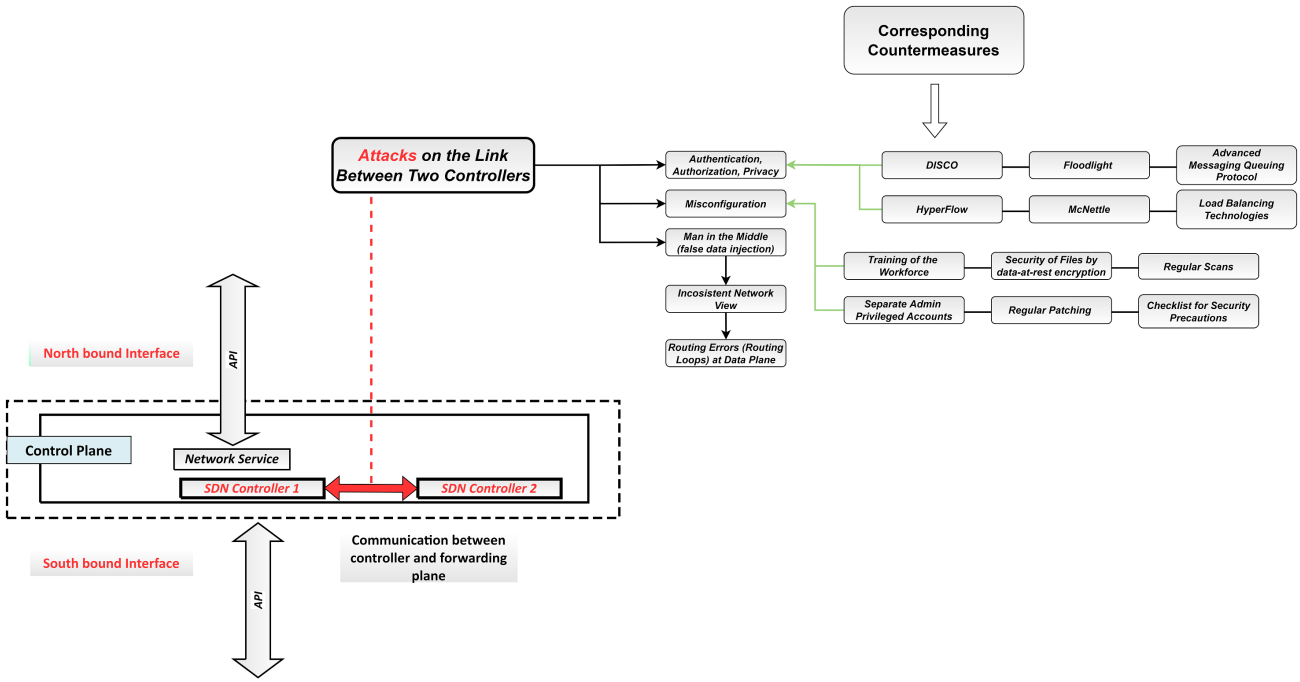

FIGURE 16. A taxonomy for the countermeasures related to the attacks between two controllers

in general because DoS/DDoS attacks are the biggest concern for SDN [27].

One of the most common malicious strategies is DDoS, which involves sending a lot of traffic in their direction to impair computer networks or resources. The primary concept behind a DoS attack is the employment of zombies dispersed over several networks or places and directed at a victim. A DoS attack's main objective is to use bandwidth and overload resources. An attack could also be carried out for other motives, such as political or financial gain or simply to disrupt services [27]. SDN offers certain unique capabilities to recognize and stop DoS threats. These include separating the control and data planes, centralizing control, programmable networking, traffic analysis capabilities, etc. Machine learning algorithms, entropy-based detection methods, and correction rate-based methods are a few available detection methods [82].

1) **Based on Entropy:** It evaluates the unpredictability of a particular attribute across a given time frame. Higher values of entropy indicate a better probability of spreading. Concentration in the distribution is represented by entropy with lower values.

2) **Based on Correction Rate:** The number of connections made and connection success rates are divided into two categories.

3) **Based on Machine Learning Algorithm:** This is widely utilized in traditional IDSs. It has been utilized for DoS detection in SDN with notable success and has been deemed successful in wired and wireless networks.

Adversary Model: An adversary model could leverage the reactive flow installation methodology of OpenFlow networks. The attacker randomly falsifies some of all sections of every packet, making it difficult to match with any prevailing flow rules in a switch. After it, SDN–aimed DDoS attack sent by the attacker with massive table–miss traffic mixed with regular traffic to its OpenFlow Switch. To process the request of the table–miss packet, the victimized switch must buffer it, and the packet will be sent with a message header, which is depicted in Figure 17. Another worst-case scenario is the OpenFlow Specifications v1.4. When the switch's memory is full, the packet in the message must contain the full packet. It could be vulnerable and exploited when attacked by flooding the network with fewer network resources.

Some common DDoS flooding attacks have been discussed below:

A. HTTP FLOOD ATTACK

The HTTP flood attack is a type of Denial-of-Service (DoS) attack that exploits the HTTP protocol to send a large number of HTTP GET requests to a server. This can result in the server becoming overwhelmed with traffic and unable to respond to legitimate requests [114].

Attack in SDN Scenario: In the context of an SDN architecture, an attacker can execute the HTTP flood attack by sending a high volume of HTTP GET requests to a specific network device, such as an SDN controller or a switch. This can cause the device to become overloaded with traffic and

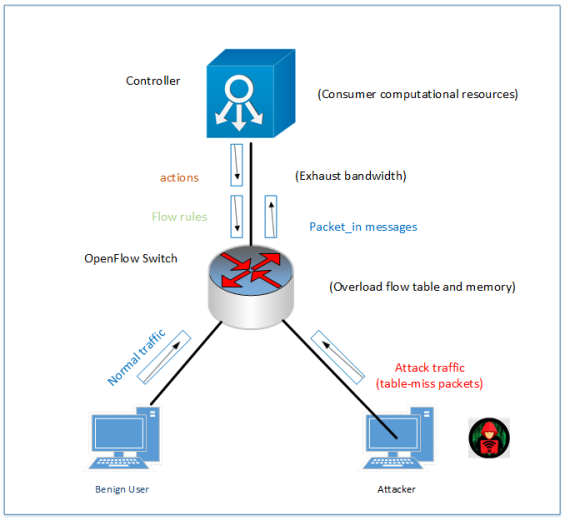

FIGURE 17. SDN aimed DoS attacks in OpenFlow networks

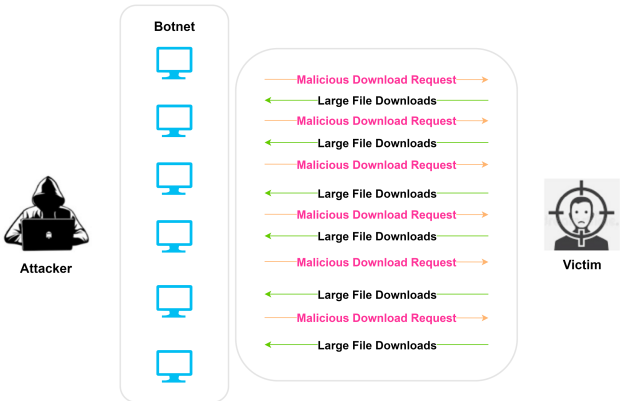

FIGURE 18. HTTP flood attack (by exploiting HTTP GET Request)

unable to process incoming requests, leading to a slowdown or even a complete shutdown of the device [115].

As shown in Figure 18, the HTTP flood attack works by sending a large number of HTTP GET requests to the target network device. These requests are typically generated using automated tools or scripts, which can quickly generate a massive volume of traffic. The attack's goal is to consume the resources of the target device, such as CPU, memory, or network bandwidth, to cause it to become unresponsive.

Possible Mitigation Strategies: To mitigate the HTTP flood attack in an SDN architecture, network administrators can employ various techniques to control the traffic flow and prevent the attacker from overwhelming the resources of the target device. Some of the mitigation techniques include:

- Rate Limiting: Implementing rate limiting mechanisms can help restrict the number of HTTP GET requests per second sent to the target device. By limiting the rate of incoming traffic, the target device can process incoming requests more efficiently and prevent a single traffic source from overwhelming the device.
- Access Control: Network administrators can ensure that only authorized users and devices can access the target device by implementing access control policies. This can help to prevent attackers from sending traffic to the device that could cause congestion and disrupt network traffic.
- Firewalling: Deploying firewalls or other security technologies that can monitor network traffic and detect and block malicious traffic that attempts to exploit the HTTP protocol.
- Intrusion Detection and Prevention: Deploying intrusion detection and prevention systems (IDS/IPS) that can detect and prevent attacks targeting the HTTP protocol can help to identify and block HTTP flood attacks in real-time.
- Load Balancing: Load balancing can help to distribute traffic across multiple devices and prevent a single device from becoming overwhelmed. This can help ensure the network remains available and responsive, even under heavy traffic conditions.
- Firmware and Software Updates: Regularly updating the firmware and software used in the SDN infrastructure can help to ensure that known vulnerabilities and exploits are addressed. This can help to prevent attackers from exploiting weaknesses in the infrastructure to launch the HTTP flood attack.

By implementing these best practices, network administrators can help mitigate the HTTP flood attack risk and ensure that the SDN infrastructure remains secure and available.

B. ICMP FLOOD ATTACK

An ICMP flood attack is a Denial-of-Service (DoS) attack that targets the Internet Control Message Protocol (ICMP), used for diagnostic and error reporting purposes in IP networks. In an ICMP flood attack, the attacker sends a large volume of ICMP echo request packets to a target network device, overwhelming the device with traffic and causing it to become unresponsive [116].

Attack in SDN Scenario: In an SDN architecture, an attacker could launch an ICMP flood attack against an SDN switch or an SDN controller. The attacker could use a tool to generate a high volume of ICMP echo request packets and send them to the target device, causing it to become overwhelmed with traffic and unable to process incoming requests [117].

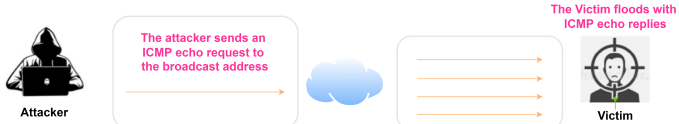

FIGURE 19. ICMP flood attack

As shown in Figure 19, the ICMP flood attack works by sending a large volume of ICMP echo request packets to the target device. These packets are sent to a specific IP address or a range of IP addresses, and they may use spoofed or legitimate source IP addresses to make it harder for network administrators to identify the source of the attack. The attack's goal is to consume the resources of the target device, such as CPU, memory, or network bandwidth, to cause it to become unresponsive.

Possible Mitigation Strategies: To mitigate the ICMP flood attack in an SDN architecture, network administrators can use several techniques to control the traffic flow and prevent the attacker from overwhelming the resources of the target device. Some of the mitigation techniques include:

- Rate Limiting: Implementing rate limiting mechanisms can help restrict the number of ICMP echo request packets per second sent to the target device. By limiting the rate of incoming traffic, the target device can process incoming requests more efficiently and prevent a single traffic source from overwhelming the device.
- Access Control: Network administrators can ensure that only authorized users and devices can access the target device by implementing access control policies. This can help to prevent attackers from sending traffic to the device that could cause congestion and disrupt network traffic.
- Firewalling: Deploying firewalls or other security technologies that can monitor network traffic and detect and block malicious traffic that attempts to exploit the ICMP protocol.
- Intrusion Detection and Prevention: Deploying intrusion detection and prevention systems (IDS/IPS) that can detect and prevent attacks targeting the ICMP protocol can help to identify and block ICMP flood attacks in real-time.
- Load Balancing: Load balancing can help to distribute traffic across multiple devices and prevent a single device from becoming overwhelmed. This can help ensure the network remains available and responsive, even under heavy traffic conditions.
- Firmware and Software Updates: Regularly updating the firmware and software used in the SDN infrastructure can help to ensure that known vulnerabilities and exploits are addressed. This can help to prevent attackers from exploiting weaknesses in the infrastructure to launch the ICMP flood attack.

By implementing these best practices, network administrators can help to mitigate the risk of the ICMP flood attack and ensure that the SDN infrastructure remains secure and available.

C. TCP SYN FLOOD ATTACK

A TCP SYN flood attack is a type of Denial-of-Service (DoS) attack that targets the TCP protocol, which is used for establishing and terminating network connections. In a TCP SYN flood attack, the attacker sends a large volume of TCP SYN requests to a target network device, overwhelming the device with traffic and causing it to become unresponsive [66].

Attack in SDN Scenario: In an SDN architecture, an attacker could launch a TCP SYN flood attack against an SDN switch or an SDN controller. The attacker could use a tool to generate a high volume of TCP SYN requests and send them to the target device, causing it to become overwhelmed with traffic and unable to process incoming requests [75].

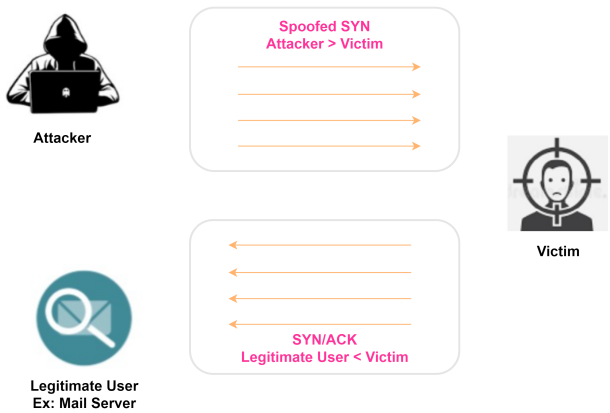

FIGURE 20. TCP SYN flood attack

As shown in Figure 20, the TCP SYN flood attack works by sending a large volume of TCP SYN requests to the target device. These requests are sent to a specific port on the target device, and they may use spoofed or legitimate source IP addresses to make it harder for network administrators to identify the source of the attack. The attack's goal is to consume the resources of the target device, such as CPU, memory, or network bandwidth, to cause it to become unresponsive.

Possible Mitigation Strategies: To mitigate the TCP SYN flood attack in an SDN architecture, network administrators can use several techniques to control the traffic flow and prevent the attacker from overwhelming the resources of the target device. Some of the mitigation techniques include:

- Rate Limiting: Implementing rate limiting mechanisms can help restrict the number of TCP SYN requests per second sent to the target device. By limiting the rate of incoming traffic, the target device can process incoming requests more efficiently and prevent a single traffic source from overwhelming the device.
- Access Control: Network administrators can ensure that only authorized users and devices can access the target device by implementing access control policies. This

can help to prevent attackers from sending traffic to the device that could cause congestion and disrupt network traffic.

- Firewalling: Deploying firewalls or other security technologies that can monitor network traffic and detect and block malicious traffic that attempts to exploit the TCP protocol.
- Intrusion Detection and Prevention: Deploying intrusion detection and prevention systems (IDS/IPS) that can detect and prevent attacks targeting the TCP protocol can help to identify and block TCP SYN flood attacks in real-time.
- Load Balancing: Load balancing can help to distribute traffic across multiple devices and prevent a single device from becoming overwhelmed. This can help ensure the network remains available and responsive, even under heavy traffic conditions.
- TCP SYN Cookies: TCP SYN cookies can help to prevent TCP SYN flood attacks by using a cryptographic algorithm to generate a unique sequence number for each TCP SYN request. This can help to ensure that only legitimate connection requests are accepted and prevent attackers from flooding the target device with bogus connection requests [118].

By implementing these best practices, network administrators can help to mitigate the risk of the TCP SYN flood attack and ensure that the SDN infrastructure remains secure and available.

D. UDP FLOOD ATTACK

A UDP flood attack is a type of Denial-of-Service (DoS) attack that targets the User Datagram Protocol (UDP), used for communication between applications on the Internet. In a UDP flood attack, the attacker sends a large volume of UDP packets to a target network device, overwhelming the device with traffic and causing it to become unresponsive [119].

Attack in SDN Scenario: In an SDN architecture, an attacker could launch a UDP flood attack against an SDN switch or an SDN controller. The attacker could use a tool to generate a high volume of UDP packets and send them to the target device, causing it to become overwhelmed with traffic and unable to process incoming requests [120], [121].

As shown in Figure 21, the UDP flood attack works by sending a large volume of UDP packets to the target device. These packets may use spoofed or legitimate source IP addresses to make it harder for network administrators to identify the source of the attack. The attack's goal is to consume the resources of the target device, such as CPU, memory, or network bandwidth, to cause it to become unresponsive.

Possible Mitigation Strategies: To mitigate the UDP flood attack in an SDN architecture, network administrators can use several techniques to control the traffic flow and

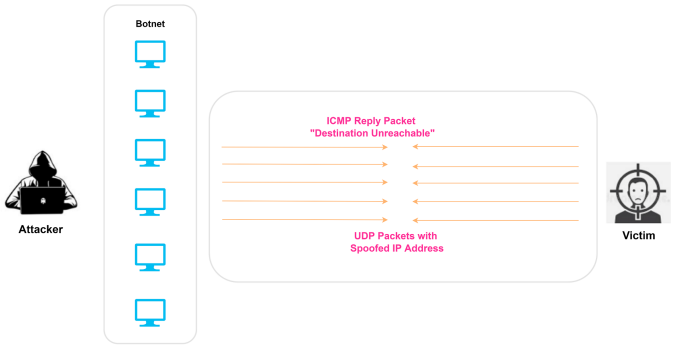

FIGURE 21. UDP flood attack

prevent the attacker from overwhelming the resources of the target device. Some of the mitigation techniques include:

- Rate Limiting: Implementing rate limiting mechanisms can help to restrict the number of UDP packets per second that are sent to the target device. By limiting the rate of incoming traffic, the target device can process incoming requests more efficiently and prevent a single traffic source from overwhelming the device.
- Access Control: Network administrators can ensure that only authorized users and devices can access the target device by implementing access control policies. This can help to prevent attackers from sending traffic to the device that could cause congestion and disrupt network traffic.
- Firewalling: Deploying firewalls or other security technologies that can monitor network traffic and detect and block malicious traffic that attempts to exploit the UDP protocol.
- Intrusion Detection and Prevention: Deploying intrusion detection and prevention systems (IDS/IPS) that can detect and prevent attacks targeting the UDP protocol can help to identify and block UDP flood attacks in real-time.
- Load Balancing: Load balancing can help to distribute traffic across multiple devices and prevent a single device from becoming overwhelmed. This can help ensure the network remains available and responsive, even under heavy traffic conditions.
- Packet Filtering: Packet filtering can filter out specific types of traffic associated with UDP flood attacks, such as packets with specific source IP addresses or destination ports.

By implementing these best practices, network administrators can help to mitigate the risk of the UDP flood attack and ensure that the SDN infrastructure remains secure and available.

E. VOIP FLOOD ATTACK

A VoIP (Voice over Internet Protocol) flood attack is a type of Denial-of-Service (DoS) attack that targets the VoIP network infrastructure. In a VoIP flood attack, the attacker sends a high volume of VoIP traffic to the target network device, overwhelming it with traffic and causing it to become unresponsive [122].

Attack in SDN Scenario: In an SDN architecture, an attacker could launch a VoIP flood attack against an SDN switch or an SDN controller. The attacker could use a tool to generate a large volume of VoIP traffic and send it to the target device, causing it to become overwhelmed and unable to process incoming requests [26].

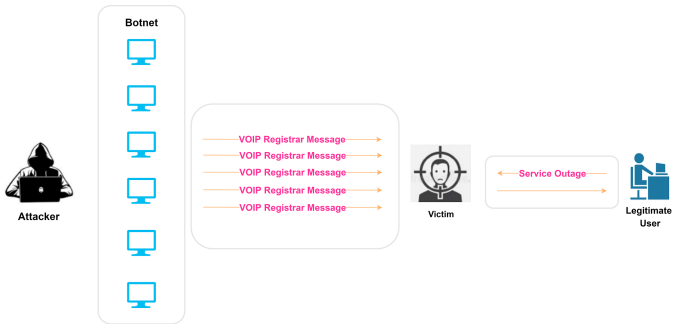

FIGURE 22. VOIP flood attack

As shown in Figure 22, a VoIP flood attack works by generating a high volume of VoIP packets and sending them to the target device. The attacker can use different types of VoIP protocols, such as Session Initiation Protocol (SIP) [123], Real-time Transport Protocol (RTP) [124], and Real-time Control Protocol (RTCP) [125] to flood the network. The attack's goal is to consume the resources of the target device, such as CPU, memory, or network bandwidth, to cause it to become unresponsive.

Possible Mitigation Strategies: To mitigate the VoIP flood attack in an SDN architecture, network administrators can use several techniques to control the flow of traffic and prevent the attacker from overwhelming the resources of the target device. Some of the mitigation techniques include:

- Rate Limiting: Implementing rate limiting mechanisms can help to restrict the number of VoIP packets per second that are sent to the target device. By limiting the rate of incoming traffic, the target device can process incoming requests more efficiently and prevent a single traffic source from overwhelming the device.
- Access Control: Network administrators can ensure that only authorized users and devices can access the target device by implementing access control policies. This can help to prevent attackers from sending traffic to the device that could cause congestion and disrupt network traffic.
- Firewalling: Deploying firewalls or other security technologies that can monitor network traffic and detect and block malicious traffic that attempts to exploit the VoIP protocol.
- Intrusion Detection and Prevention: Deploying intrusion detection and prevention systems (IDS/IPS) that can detect and prevent attacks targeting the VoIP protocol can help to identify and block VoIP flood attacks in real-time.
- Quality of Service (QoS): Implementing QoS can help to prioritize VoIP traffic over other types of network traffic. This can help to ensure that VoIP traffic receives the necessary network resources to operate efficiently.

By implementing these best practices, network administrators can help to mitigate the risk of the VoIP flood attack and ensure that the SDN infrastructure remains secure and available.

The effects of a DoS attack on modified SDN layers or mechanisms are briefly described below -

1) **Application Layer:** Installing a new application in an SDN system is fairly simple due to the fundamentals of centralized design. It is more vulnerable to security concerns because of its centralized form, including access control, policy violations, a conflict with flow rules, etc. The application may transgress security-related rules while retrieving network and packet-related information from the controller. A malicious application might install Unauthorized flow rules to interfere with normal operation.
2) **Data forwarding / Infrastructure Layer:** The OpenFlow switch is the first component a DoS/DDoS attack affects. The header of an arriving packet is compared to the flow rules kept in flow tables. The switch will respond by the matched rule if a match is discovered. The switch will then pass the packet to the controller (as per the established policy) through the control channel by the OpenFlow agent installed in the switch if there is no match with the flow tables. The remaining data portion of the packet stays kept in the OpenFlow switch's packet buffer region if the switch just forwards the packet header. The OpenFlow switch has a finite amount of processor and memory resources. If a host connected to the switch repeatedly sends packets with different source and destination addresses, the switch will look up each packet it gets in the associated flow table and, if it finds it there, forward it to the controller. In this scenario, the controller will add a flow rule for each packet to the flow table of the relevant switch, which may result in flow table overflow. In addition, resource limitations may cause the OpenFlow agent to become overwhelmed. If the packet buffer fills up, the entire packet will be forwarded to the controller; however, if the OpenFlow agent is overloaded, it won't

be able to send additional packets, and hence the packet will be lost.

3) **Flow Table:** Any rule change from the controller is dropped if the flow table to the switch is full since there is no room for it. The switch won't forward any additional packets until free space in the flow table becomes available. Any host connected to the impacted switch cannot deliver the necessary level of service. There may be packet loss from sources, and latency will be considerable.

4) **Control Layer (Controller):** It includes the controller, a crucial part of the SDN system. The controller is in charge of giving the OpenFlow switches the relevant data when they ask for it. The controller might be a potential target for an attacker because it is the center point of failure. One crucial feature offered by the SDN controller is the installation of switches' flow rules. Sending many packet-in messages and requesting a lot of flow rules can make a controller's resources dysfunctional. As a result, responding to inquiries takes longer. A switch attack will only have a local impact, whereas a successful DoS assault on the controller will impact the entire infrastructure.

5) **Control Channel:** The control channel is the communication channel between the controller and the OpenFlow switch. There are two ways to do it: out-of-band, where the link is made using a special physical link with the controller, and in-band, where the link is created utilizing the current data plane connections. Due to a DoS attack, these links' limited capacity could be negatively impacted. Links in the in-band category will be more impacted because the physical link is used by regular traffic.

VII. RESEARCH GAPS

While finding the countermeasures for the attacks against the control plane, we also got some of the following research gaps or future research prospects.

A. ENTROPY METHOD

While entropy-based methods have shown promise in detecting and defending against multiple attacks in SDN, several research gaps and limitations must be addressed to enhance their effectiveness. These gaps hinder the accuracy, scalability, and reliability of current approaches. The following issues have been identified:

1) **False-positive rate increase:** Existing entropy-based methods experience an increase in the false-positive rate when different attack traffic rates are considered [126]. This indicates a pressing need to improve the precision of detection mechanisms to avoid unnecessary alarms and enable efficient threat identification.

2) **Low attack detection rate for low-rate attacks:** Despite their effectiveness in detecting high-rate attacks, entropy-based methods often exhibit a low detection rate for low-rate attack traffic targeting multiple victims [126]. This limitation compromises the network's ability to identify and mitigate subtle and sophisticated attacks, posing a significant security risk.

3) **Delay in packet processing:** The processing delay in entropy-based methods can be detrimental to timely attack detection and mitigation [126]. The time required to analyze and respond to incoming packets introduces a vulnerability window during which attacks can potentially cause damage.

4) **Fixed threshold reliance:** Many entropy-based approaches rely on fixed thresholds to detect different DDoS attack traffic rates targeting single or multiple hosts [127]. This rigid threshold reliance makes it challenging to effectively identify low-rate DDoS attacks with a high detection rate and low false-positive rate, demanding more flexible and adaptive detection mechanisms.

5) **Single point of controller failure:** The centralized controller in SDN architectures presents a single failure point, impacting entropy-based methods' resilience and reliability [128]. A successful attack on the controller can paralyze the entire network, emphasizing the importance of distributed and fault-tolerant architectures.

6) **High controller performance overhead:** Deploying entropy-based methods can impose a substantial performance overhead on the SDN controller [128]. This overhead may hinder the controller's ability to handle network traffic, compromising its responsiveness and scalability efficiently.

7) **Threshold selection impact:** Selecting an appropriate threshold is critical for the effectiveness of entropy-based methods [128]. An incorrect threshold choice can lead to compromised detection accuracy, either missing attacks or generating a high number of false positives.

Limitations in adaptive timeout mechanisms for entropy-based methods:

1) **Impact of probing accuracy:** The accuracy of probing, involving Round-Trip Time (RTT) measurement for testing packets, directly influences the effectiveness of attack detection [129] [130]. Inaccurate probing can hinder precisely identifying and responding to attacks, necessitating improved probing techniques.

2) **Consideration of general timeout assignments:** Existing adaptive timeout mechanisms for flow rule state transitions primarily rely on general timeout assignments [129] [130]. This assumption overlooks the influence of packet inter-arrival times, a crucial feature that impacts the accuracy and efficiency of adaptive timeout mechanisms.

These identified research gaps and limitations highlight

the necessity for further investigation and advancements in entropy-based methods in SDN. Addressing these challenges makes enhancing attack detection mechanisms' accuracy, reliability, and efficiency possible, thereby fortifying SDN networks against various threats.

B. STATISTICAL ANALYSIS-BASED ANOMALY DETECTION METHOD

While the statistical analysis-based anomaly detection method has gained popularity for detecting DDoS attacks in SDN, several research gaps and limitations must be addressed to enhance its effectiveness in real-world scenarios. The following issues have been identified:

1) **Limited by a fixed threshold:** The statistical analysis method relies on a single fixed threshold for distinguishing regular and attack traffic [131]. However, this fixed threshold approach can result in misjudgments and false positives, as it may not account for variations in attack patterns and network conditions.
2) **Difficulty in threshold adjustment:** Setting an optimal threshold value for statistical analysis detection methods poses challenges, as the appropriate threshold may vary across different network environments [131]. This adjustment process requires specialized knowledge and expertise, limiting its applicability to practitioners without extensive experience.
3) **Lack of reliability in real-world networks:** Relying solely on statistical analysis for identifying abnormal traffic may not provide reliable results in realistic network environments [131]. The method's reliance on statistical patterns and fixed thresholds can lead to false detection, particularly when faced with evolving attack techniques and dynamic network conditions.
4) **Need for robust and adaptable detection mechanisms:** Addressing the limitations of fixed thresholds requires the development of more robust and adaptable detection mechanisms within the statistical analysis-based approach [131]. These mechanisms should dynamically adjust thresholds based on the specific network environment, evolving attack techniques, and patterns of abnormal traffic.
5) **Consideration of multiple statistical properties:** While statistical analysis methods commonly focus on a single or a few statistical properties, exploring the effectiveness of considering multiple statistical properties simultaneously is crucial [131]. Such an approach could capture a broader range of deviations from normal traffic behavior, enhancing the accuracy and reliability of anomaly detection.

These identified research gaps highlight the necessity for further investigation and advancements in statistical analysis-based anomaly detection methods for DDoS attacks in SDN. Addressing these challenges makes enhancing the reliability, adaptability, and accuracy of attack detection mechanisms possible, thereby fortifying SDN networks against various threats.

C. RELAY LINK FORGED ATTACK

Relay link forged attacks pose challenges in real-world data collection and model training environments. Simulated environments also have limitations, such as overfitting risks and the inability to capture real-world complexity and diversity [132]. To address these gaps, the following strategies and research areas should be explored:

1) **Real-world data collection challenges:** Incompletely controlled real environments introduce the risk of malicious samples in the collected data, which can impact the effectiveness of attack detection model training [132]. Research is needed to develop robust techniques to mitigate the presence of malicious samples and ensure the reliability of training datasets.
2) **Overfitting risks in simulated environments:** Simulated environments used for dataset collection may not fully generalize results in real-world scenarios, leading to overfitting risks [132]. There is a need to explore techniques to minimize overfitting, such as employing diverse datasets from various simulated environments and cross-validating the results.
3) **Performance degradation in real-world deployments:** Simulated datasets may not accurately capture the complexity and diversity of real-world situations, potentially resulting in performance degradation when deploying detection models in practical SDN environments [132]. Further research is needed to bridge the gap between simulated and real-world datasets, ensuring the models perform effectively and reliably in real-world scenarios.
4) **Generalizability of detection models:** To enhance the generalizability of attack detection models, using multiple datasets collected from various simulation environments is recommended [132]. This approach enables cross-validation and improves the model's ability to detect relay link forged attacks across different scenarios and settings.
5) **Validation in actual SDN engineering environments:** While experiments are conducted in simulated environments, there is a need to verify and deploy the developed approaches in real SDN engineering environments [132]. This step is crucial to assess the model's performance, reliability, and suitability for practical deployment against relay link forged attacks.

These research gaps highlight the importance of effectively addressing challenges in data collection, model training, overfitting, performance degradation, generalizability, and real-world deployment to detect and mitigate relay link forged attacks in SDN.

D. TOPOLOGY FORGERY ATTACKS IN SDN

Topology forgery attacks in SDN encompass protocol-based attacks that exploit vulnerabilities without direct access to the control plane or knowledge of controller weaknesses. These attacks pose security threats through link forgery attacks (LFA) and host location hijacking attacks (HLHA). The specific RLFA (Relay Link Forgery Attack) studied by Alhaj et al. [31] falls under LFA.

1) **Ineffectiveness of existing defense mechanisms:** Existing defense mechanisms, such as Topo Guard proposed in the article, aim to protect against LFA by implementing port label policies [31]. However, a research gap exists in these mechanisms' effectiveness, as adversaries can disrupt host labels and impersonate switches using relay LLDP packets, rendering the defense mechanisms inadequate to defend against RLFA [31].
2) **Mitigation of RLFA attacks:** Research is needed to develop effective mitigation techniques specifically targeting RLFA attacks within the LFA category [31]. The goal is to enhance the ability to detect and prevent the introduction of false links between switches in the controller's topology view, ensuring the integrity and accuracy of the network's perceived structure.
3) **Detection and prevention of HLHA:** HLHA, which involves manipulating the position of hosts to mislead network traffic, poses significant security risks. Further research is required to develop robust detection and prevention mechanisms to accurately identify and mitigate HLHA attacks within SDN environments.
4) **Improving security in the control plane:** Since topology forgery attacks exploit vulnerabilities without accessing the control plane, it is necessary to enhance its security. Research should focus on identifying and addressing vulnerabilities in the control plane to prevent unauthorized manipulation of the network's topology and protect against topology forgery attacks.

These research gaps highlight the need for further investigation and advancements in detecting, preventing, and mitigating topology forgery attacks, with a particular focus on RLFA and HLHA. Addressing these gaps will contribute to developing more secure and resilient SDN architectures.

E. SDN CONTROLLER PLACEMENT

SDN controller placement plays a crucial role in optimizing network performance and minimizing latency. However, several research gaps and limitations can be identified in the existing literature, as outlined below:

1) **Lack of consideration for DDoS attacks:** Previous studies, such as Haque et al. [133], have focused on optimizing controller placement based on latency and response time but have not specifically addressed the placement strategy under DDoS attacks. This research gap highlights the need for investigations into the impact of DDoS attacks on controller placement and the development of robust placement strategies that account for attack scenarios.
2) **Limited exploration of DDoS attack defense mechanisms:** While Haque et al. [133] proposed a DDoS blocking system using the OpenFlow interface, a research gap exists in exploring comprehensive defense mechanisms against DDoS attacks in SDN. Future research should aim to develop more sophisticated and adaptive defense techniques that consider factors such as attack detection, mitigation, and response to ensure the resilience of SDN networks against DDoS attacks.
3) **Evaluation of reliability and scalability:** Haque et al. [133] emphasized the need to enhance the reliability of SDN controllers using heuristic algorithms. However, there is a research gap in evaluating the scalability and effectiveness of these algorithms in large-scale SDN deployments. Further research is required to assess the performance and efficiency of controller placement strategies under various network sizes and traffic loads.
4) **Bridging the gap between theory and practice:** Although Haque et al. [133] proposed an enhanced model for SDN controller placement, validating and deploying these approaches in real-world SDN environments is necessary. Practical deployment considerations, such as compatibility with existing network infrastructures and integration challenges, must be addressed to bridge the gap between theoretical advancements and their implementation in production networks.
5) **Comprehensive understanding of DDoS attack characteristics:** While Haque et al. [133] discussed DDoS attack trends and characteristics in cloud computing environments, there is a research gap in understanding evolving DDoS attack techniques and their implications for SDN. Future research should focus on identifying emerging attack vectors, their impact on SDN networks, and developing countermeasures to mitigate their effects.

These research gaps highlight the necessity for further investigation and advancements in SDN controller placement, specifically addressing the challenges posed by DDoS attacks, evaluating reliability and scalability, bridging the theory-practice gap, and gaining a comprehensive understanding of DDoS attack characteristics.

F. OPTIMAL SDN DEPLOYMENT

Optimal SDN deployment involves selecting suitable controller platforms and assessing the performance of SDN OpenFlow controllers. However, several research gaps and limitations can be identified in the existing literature, as outlined below:

1) **Lack of comprehensive assessment criteria:** Existing research, including the work by Badotra et al. [134], has surveyed and examined performance assessment criteria for OpenFlow controllers in SDNs. However, there is a research gap in developing comprehensive

and standardized assessment criteria that consider diverse network conditions, parameters, metrics, and scaling of network load resources. Future research should focus on defining a comprehensive set of criteria to evaluate SDN controllers' performance effectively.

2) **Limited analysis of multiple controller scenarios:** While Badotra et al. [134] explored performance analysis of OpenFlow-based SDN controllers conducted by different scholars and in groups of two or more controllers, there is a research gap in analyzing larger-scale scenarios with multiple controllers. Further research is needed to assess the performance and scalability of SDN deployments that involve multiple controllers under various network conditions and load scenarios.
3) **Understanding the impact of network topology design:** The varying network topology design is an essential aspect of SDN deployment. However, there is a research gap in understanding the influence of different network topologies on the performance of SDN controllers. Future research should investigate the relationship between network topology design and controller performance to optimize SDN deployments for various network configurations.
4) **Standardization of performance evaluation:** SDN controllers need standardization in performance evaluation methodologies to enable meaningful comparisons across different studies. Currently, there is a lack of consensus on evaluation techniques, metrics, and experimental setups. Addressing this research gap would facilitate more reliable and consistent performance evaluations of SDN controllers.

These research gaps highlight the need for further investigation and advancements in optimal SDN deployment, including developing comprehensive assessment criteria, analyzing multiple controller scenarios, understanding the impact of network topology design, and standardization of performance evaluation methodologies.

G. SDN-ESRC

Software-Defined Networking Endogenously Secure Resilient Control (SDN-ESRC) is proposed by Ren et al. [135] as a resilient and endogenously secure control plane for SDN. While the concept shows promise, there are research gaps and limitations in its implementation, as outlined below:

1) **Handling multiple heterogeneous controllers:** SDN-ESRC utilizes a variety of heterogeneous controllers, such as RYU, Open Daylight, and ONOS, to build the control plane. However, a research gap exists in effectively managing and coordinating the interactions between these controllers. Further research is needed to explore efficient strategies for handling multiple controllers and ensuring seamless communication and cooperation among them.
2) **Impact on network update time:** Using multiple controllers in SDN-ESRC may increase the time required to bring the network up to date. This can potentially impact network responsiveness and agility, particularly in dynamic environments where rapid network updates are necessary. Future research should focus on minimizing the update time and optimizing the synchronization process when employing multiple controllers.
3) **Achieving a high degree of controlled security:** SDN-ESRC aims to provide endogenous security for the SDN control plane. However, a research gap exists in guaranteeing a very high level of controlled security when employing a range of heterogeneous controllers. Further research is needed to develop comprehensive security mechanisms and coordination strategies to ensure robust security across all controller instances.
4) **Coordination challenges and complexity:** Managing multiple heterogeneous controllers introduces coordination challenges and complexity in the control plane. Research is required to explore efficient mechanisms for dynamically and adaptively selecting controller instances, identifying and repairing control message errors, and maintaining overall control plane integrity. Addressing these challenges will enhance the reliability and effectiveness of SDN-ESRC.

These research gaps highlight the need for further investigation and advancements in implementing SDN-ESRC, particularly in managing multiple heterogeneous controllers, minimizing network update time, achieving a high level of controlled security, and addressing coordination challenges and complexity.

H. FLOW-TABLE FLOODING

Flow-table flooding attacks pose significant challenges to SDN networks [31]. Understanding and mitigating these attacks require further research, as outlined below:

1) **Impact on controller performance:** Flow-table flooding attacks overwhelm the flow table with a large volume of packets, which can significantly impact the performance of the SDN controller. Research is needed to understand the specific performance bottlenecks caused by flow-table flooding and develop efficient mitigation techniques to alleviate the strain on the controller's processing capabilities.
2) **Flow table management and scalability:** Flow tables in SDN networks have limited capacity, making them susceptible to overflow during flow-table flooding attacks. Addressing this research gap involves exploring approaches to improve flow table management and scalability, allowing the network to handle a larger volume of flows while maintaining efficient packet processing.
3) **Detection and mitigation strategies:** Detecting flow-table flooding attacks in real-time is crucial for timely

mitigation. Research is needed to develop effective detection algorithms that can identify flow-table flooding attacks accurately and promptly. Additionally, mitigation strategies should be explored to dynamically adapt the flow table and prioritize critical flows during an attack, ensuring the controller's performance is not severely impacted.

4) **Optimal flow table size determination:** Determining the optimal size of the flow table is essential to balance the network's performance and security. Research should focus on identifying the factors that influence the appropriate size of the flow table, such as network size, traffic patterns, and application requirements. This will help design SDN architectures with appropriately sized flow tables that can withstand flow-table flooding attacks without compromising performance.

These research gaps highlight the need for further investigation and advancements in understanding flow-table flooding attacks, improving flow-table management and scalability, developing efficient detection and mitigation strategies, and determining optimal flow-table sizes to enhance the resilience and performance of SDN networks.

VIII. FUTURE WORK

In order to address the evolving landscape of SDN security, our research aims to continually advance the field by identifying emerging threats and proposing effective countermeasures. The following outlines our planned future work, which focuses on enhancing threat detection, developing dynamic defense strategies, and leveraging advanced techniques to strengthen SDN security.

Some points describing the key areas of focus have been described below:

1) **Comprehensive threat identification:** We will expand our research to encompass a broader range of threats and vulnerabilities specific to SDN environments. This includes analyzing emerging attack vectors and understanding their potential impact on network security.
2) **Development of countermeasures and algorithms:** Our future work will involve the development of innovative countermeasures and algorithms to mitigate identified threats. We will explore novel approaches that enhance the resilience and security of SDN controllers and networks.
3) **Integration of deep learning and machine learning:** We plan to integrate deep learning and machine learning techniques into our security framework to improve threat detection and response. This includes exploring simulation-based or dataset-based training to enhance real-time attack detection and automate adaptive network rule adjustments.
4) **Dynamic network rule updates:** We recognize the importance of adapting network rules in response to detected threats. Our future work will focus on developing strategies to dynamically update network rules based on real-time threat intelligence, ensuring effective mitigation and minimizing the impact of attacks.
5) **Research validation and refinement:** Thorough research and validation will be conducted before implementing our proposed approaches. We will carefully select appropriate algorithms and datasets to emulate various attack scenarios and validate the effectiveness of our defense strategies.
6) **Continuous research updates:** As we progress, we will continuously update our research findings to reflect the latest findings and advancements in SDN security. This ensures that our work remains relevant and provides valuable insights for the broader research community.

These future research directions underline our commitment to advancing SDN security by addressing emerging threats, developing effective countermeasures, leveraging advanced techniques, and continuously updating our research to reflect the latest developments in the field.

IX. CONCLUSION

This research paper has delved into the realm of Software-Defined Networking (SDN) security, with a specific focus on the control plane. Throughout our investigation, we have made some contributions that enhance the understanding of SDN security and provide practical guidance for researchers and practitioners in the field.

Firstly, we have established a comprehensive attack classification and taxonomy, which organizes the various attacks targeting the SDN control plane based on their specific attack surfaces. This structured classification offers a deeper understanding of SDN environments' attack vectors, facilitating more effective security measures.

Moreover, we have presented a unique taxonomical representation of the identified attacks, offering a systematic framework for analyzing and comprehending their characteristics. This taxonomical approach enables researchers and practitioners to gain a clear and organized view of the attacks, allowing for a better understanding of the relationships between different attack types. Additionally, we have proposed countermeasure taxonomies aligned with the attack taxonomy. These countermeasures provide practical guidance for implementing effective security measures in SDN environments. By aligning the countermeasures with the corresponding attack categories, we offer a targeted and proactive approach to mitigating or preventing these attacks.

Lastly, our research paper conducts a thorough research gap analysis, identifying limitations and research needs in SDN security. By highlighting these gaps, we provide valuable insights for future researchers, guiding them toward potential research directions and enabling them to address the current shortcomings in the field.

Overall, the contributions made in this research paper significantly enhance the understanding of attacks against SDN control planes. The classification and taxonomy of attacks, taxonomical representation, countermeasure taxonomies, and research gap analysis collectively contribute to the body of knowledge on SDN security. This research serves as a foundation for future research and development in the field, enabling the implementation of robust security measures and the advancement of SDN technology.

As the field of SDN continues to evolve, researchers and practitioners must remain vigilant and proactive in addressing security challenges. By leveraging the insights and recommendations provided in this paper, stakeholders can fortify their SDN deployments, ensuring their network infrastructure's integrity, availability, and confidentiality.

REFERENCES

[1] Lily Yang, Ram Dantu, Terry Anderson, and Ram Gopal. Forwarding and control element separation (forces) framework (no. rfc3746). Technical report, 2004.

[2] TV Lakshman, T Nandagopal, Ramachandran Ramjee, K Sabnani, and T Woo. The softrouter architecture. In Proc. ACM SIGCOMM Workshop on Hot Topics in Networking, volume 2004, 2004.

[3] J Salim, H Khosravi, Andi Kleen, and Alexey Kuznetsov. Linux netlink as an ip services protocol (no. rfc3549). Technical report, 2003.

[4] Martin Casado, Michael J Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. Ethane: Taking control of the enterprise. ACM SIGCOMM computer communication review, 37(4):1–12, 2007.

[5] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. ACM SIGCOMM computer communication review, 38(2):69–74, 2008.

[6] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. Nox: towards an operating system for networks. ACM SIGCOMM computer communication review, 38(3):105–110, 2008.

[7] ONF. Software-Defined Networking (SDN) Definition.

[8] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn: an intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review, 44(2):87–98, 2014.

[9] Danda B Rawat and Swetha R Reddy. Software defined networking architecture, security and energy efficiency: A survey. IEEE Communications Surveys & Tutorials, 19(1):325–346, 2016.

[10] Andreas Voellmy, Hyojoon Kim, and Nick Feamster. Procera: A language for high-level reactive network control. In Proceedings of the first workshop on Hot topics in software defined networks, pages 43–48, 2012.

[11] Rob Enns, Martin Bjorklund, Juergen Schoenwaelder, and Andy Bierman. Network configuration protocol (netconf) (rfc6241). Technical report, 2011.

[12] AbdelRahman Abdou, Paul C Van Oorschot, and Tao Wan. Comparative analysis of control plane security of sdn and conventional networks. IEEE Communications Surveys & Tutorials, 20(4):3542–3559, 2018.

[13] Philip Porras, Seungwon Shin, Vinod Yegneswaran, Martin Fong, Mabry Tyson, and Guofei Gu. A security enforcement kernel for openflow networks. In Proceedings of the first workshop on Hot topics in software defined networks, pages 121–126, 2012.

[14] Kapil Dhamecha and Bhushan Trivedi. Sdn issues-a survey. International Journal of Computer Applications, 73(18), 2013.

[15] Ben Pfaff, Justin Pettit, Keith Amidon, Martin Casado, Teemu Koponen, and Scott Shenker. Extending networking into the virtualization layer. In Hotnets, 2009.

[16] Directory. Indigo Virtual Switch (IVS), sdxcentral, 2023.

[17] Pica8. PicOS: Delivering networking freedom, Pica8, 2023.

[18] Andreas Voellmy and Paul Hudak. Nettle: Taking the sting out of programming network routers. In International Symposium on Practical Aspects of Declarative Languages, pages 235–249. Springer, 2011.

[19] Yiannis Yiakoumis, Julius Schulz-Zander, and Jiang Zhu. Pantou : OpenFlow 1.0 for OpenWRT, 2011.

[20] Lin-du Aaronshang, chunyiliao. Pica8 Xorplus, 2016.

[21] Fei Hu, Qi Hao, and Ke Bao. A survey on software-defined network and openflow: From concept to implementation. IEEE Communications Surveys & Tutorials, 16(4):2181–2206, 2014.

[22] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1):14–76, 2014.

[23] William Stallings. Software-defined networks and openflow. The internet protocol Journal, 16(1):2–14, 2013.

[24] Adrian Lara, Anisha Kolasani, and Byrav Ramamurthy. Network innovation using openflow: A survey. IEEE communications surveys & tutorials, 16(1):493–512, 2013.

[25] Bruno Astuto A Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications surveys & tutorials, 16(3):1617–1634, 2014.

[26] Shi Dong, Khushnood Abbas, and Raj Jain. A survey on distributed denial of service (ddos) attacks in sdn and cloud computing environments. IEEE Access, 7:80813–80828, 2019.

[27] Jagdeep Singh and Sunny Behal. Detection and mitigation of ddos attacks in sdn: A comprehensive review, research challenges and future directions. Computer Science Review, 37:100279, 2020.

[28] Muhammad Arif, Guojun Wang, Oana Geman, Valentina Emilia Balas, Peng Tao, Adrian Brezulianu, and Jianer Chen. Sdn-based vanets, security attacks, applications, and challenges. Applied Sciences, 10(9):3217, 2020.

[29] Juan Camilo Correa Chica, Jenny Cuatindioy Imbachi, and Juan Felipe Botero Vega. Security in sdn: A comprehensive survey. Journal of Network and Computer Applications, 159:102595, 2020.

[30] Felipe S Dantas Silva, Esau Silva, Emidio P Neto, Marcilio Lemos, Augusto J Venancio Neto, and Flavio Esposito. A taxonomy of ddos attack mitigation approaches featured by sdn technologies in iot scenarios. Sensors, 20(11):3078, 2020.

[31] Ali Nadim Alhaj and Nitul Dutta. Analysis of security attacks in sdn network: A comprehensive survey. In Hiren Kumar Deva Sarma, Valentina Emilia Balas, Bhaskar Bhuyan, and Nitul Dutta, editors, Contemporary Issues in Communication, Cloud and Big Data Analytics, pages 27–37, Singapore, 2022. Springer Singapore.

[32] V Thirupathi, CH Sandeep, Naresh Kumar, and P Pramod Kumar. A comprehensive review on sdn architecture, applications and major benifits of sdn. International Journal of Advanced Science and Technology, 28(20):607–614, 2019.

[33] Ying-Dar Lin, Po-Ching Lin, Chih-Hung Yeh, Yao-Chun Wang, and Yuan-Cheng Lai. An extended sdn architecture for network function virtualization with a case study on intrusion prevention. IEEE Network, 29(3):48–53, 2015.

[34] Christopher Janz, Lyndon Ong, Karthik Sethuraman, and Vishnu Shukla. Emerging transport sdn architecture and use cases. IEEE Communications Magazine, 54(10):116–121, 2016.

[35] Karthik Raghunath and Prabhakar Krishnan. Towards a secure sdn architecture. In 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pages 1–7. IEEE, 2018.

[36] Krzysztof Cabaj, Jacek Wytrebowicz, Slawomir Kuklinski, Pawel Radziszewski, and Khoa Truong Dinh. Sdn architecture impact on network security. In FedCSIS (Position Papers), pages 143–148, 2014.

[37] Seyed Mohammad Mousavi and Marc St-Hilaire. Early detection of ddos attacks against sdn controllers. In 2015 international conference on computing, networking and communications (ICNC), pages 77–81. IEEE, 2015.

[38] Kevin Benton, L Jean Camp, and Chris Small. Openflow vulnerability assessment. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pages 151–152, 2013.

[39] Deyun Gao, Zehui Liu, Ying Liu, Chuan Heng Foh, Ting Zhi, and Han-Chieh Chao. Defending against packet-in messages flooding attack under sdn context. Soft Computing, 22:6797–6809, 2018.

[40] Longyan Ran, Yunhe Cui, Chun Guo, Qing Qian, Guowei Shen, and Huanlai Xing. Defending saturation attacks on sdn controller: A confusable instance analysis-based algorithm. Computer Networks, 213:109098, 2022.

[41] Samer Khamaiseh, Edoardo Serra, Zhiyuan Li, and Dianxiang Xu. Detecting saturation attacks in sdn via machine learning. In 2019 4th

International Conference on Computing, Communications and Security (ICCCS), pages 1–8. IEEE, 2019.

[42] Xuanbo Huang, Kaiping Xue, Yitao Xing, Dingwen Hu, Ruidong Li, and Qibin Sun. Fsdm: Fast recovery saturation attack detection and mitigation framework in sdn. In 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pages 329–337. IEEE, 2020.

[43] Qi Li, Jiahao Cao, Mingwei Xu, and Kun Sun. Flow Table Overflow Attacks, pages 1–3. Springer International Publishing, Cham, 2019.

[44] Aditya Patwardhan, Deepthi Jayarama, Nitish Limaye, Shivaji Vidhale, Zarna Parekh, and Khaled Harfoush. Sdn security: Information disclosure and flow table overflow attacks. In 2019 IEEE Global Communications Conference (GLOBECOM), pages 1–6. IEEE, 2019.

[45] Seungwon Shin, Yongjoo Song, Taekyung Lee, Sangho Lee, Jaewoong Chung, Phillip Porras, Vinod Yegneswaran, Jiseong Noh, and Brent Byunghoon Kang. Rosemary: A robust, secure, and high-performance network operating system. In Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, pages 78–89, 2014.

[46] Christian Röpke and Thorsten Holz. Retaining control over sdn network services. In 2015 International Conference and Workshops on Networked Systems (NetSys), pages 1–5. IEEE, 2015.

[47] Christian Röpke and Thorsten Holz. On network operating system security [rh]. International Journal of Network Management, 26(1):6–24, 2016.

[48] Christian Röpke. Sdn malware: problems of current protection systems and potential countermeasures. Sicherheit 2016-Sicherheit, Schutz und Zuverlässigkeit, 2016.

[49] Scott Hogg. SDN Security Attack Vectors and SDN Hardening, 2014.

[50] Yash Sinha, K Haribabu, et al. A survey: Hybrid sdn. Journal of Network and Computer Applications, 100:35–55, 2017.

[51] Leonard Richardson and Sam Ruby. RESTful web services. " O'Reilly Media, Inc.", 2008.

[52] Balakrishnan Chandrasekaran and Theophilus Benson. Tolerating sdn application failures with legosdn. In Proceedings of the 13th ACM workshop on hot topics in networks, pages 1–7, 2014.

[53] Vinod Yegneswaran Phillip Porras, Steven Cheung, Martin Fong, Keith Skinner. Securing the Software-Defined Network Control Layer. In NDSS Symposium 2015, San Diego, California, 2015. NDSS Symposium.

[54] Dennis Tatang, Florian Quinkert, Joel Frank, Christian Röpke, and Thorsten Holz. Sdn-guard: Protecting sdn controllers against sdn rootkits. In 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pages 297–302. IEEE, 2017.

[55] Paul Congdon. Link layer discovery protocol and mib. V1. 0 May, 20(2002):1–20, 2002.

[56] Sonali Sen Baidya and Rattikorn Hewett. Link discovery attacks in software-defined networks: Topology poisoning and impact analysis. J. Commun., 15(8):596–606, 2020.

[57] Tri-Hai Nguyen and Myungsik Yoo. Analysis of link discovery service attacks in sdn controller. In 2017 International Conference on Information Networking (ICOIN), pages 259–261. IEEE, 2017.

[58] Sungmin Hong, Lei Xu, Haopei Wang, and Guofei Gu. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In Ndss, volume 15, pages 8–11, 2015.

[59] A OpenDaylight. Opendaylight, a linux foundation collaborative project. 2013.

[60] Christian Röpke and Thorsten Holz. Sdn rootkits: Subverting network operating systems of software-defined networks. In Herbert Bos, Fabian Monrose, and Gregory Blanc, editors, Research in Attacks, Intrusions, and Defenses, pages 339–356, Cham, 2015. Springer International Publishing.

[61] Jeffrey C Mogul and Paul Congdon. Hey, you darned counters! get off my asic! In Proceedings of the first workshop on Hot topics in software defined networks, pages 25–30, 2012.

[62] Christian Röpke. Sdn ro2tkits: A case study of subverting a closed source sdn controller. SICHERHEIT 2018, 2018.

[63] Ahmad Aseeri, Nuttapong Netjinda, and Rattikorn Hewett. Alleviating eavesdropping attacks in software-defined networking data plane. In Proceedings of the 12th Annual Conference on Cyber and Information Security Research, pages 1–8, 2017.

[64] Changhoon Yoon, Seungsoo Lee, Heedo Kang, Taejune Park, Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. Flow wars: Systemizing the attack surface and defenses in software-defined networks. IEEE/ACM Transactions on Networking, 25(6):3514–3530, 2017.

[65] Neil Long and Rob Thomas. Trends in denial of service attack technology. CERT Coordination Center, 648(651):569, 2001.

[66] Wesley Eddy. Tcp syn flooding attacks and common mitigations. Technical report, 2007.

[67] Ahmed M Abdelmoniem and Brahim Bensaou. The switch from conventional to sdn: The case for transport-agnostic congestion control. arXiv preprint arXiv:2209.04729, 2022.

[68] Seonhyeok Kim, Jaehyeok Son, Ashis Talukder, and Choong Seon Hong. Congestion prevention mechanism based on q-leaning for efficient routing in sdn. In 2016 International Conference on Information Networking (ICOIN), pages 124–128. IEEE, 2016.

[69] M Saravanan, Arud Selvan Sundaramurthy, Divya Sundar, and K Hiba Sadia. Extending sdn framework for communication networks. In Internet of Things. IoT Infrastructures: Second International Summit, IoT 360° 2015, Rome, Italy, October 27-29, 2015, Revised Selected Papers, Part II, pages 539–550. Springer, 2016.

[70] Ricardo Macedo, Rafael de Castro, Aldri Santos, Yacine Ghamri-Doudane, and Michele Nogueira. Self-organized sdn controller cluster conformations against ddos attacks effects. In 2016 ieee global communications conference (globecom), pages 1–6. IEEE, 2016.

[71] Zhaogang Shu, Jiafu Wan, Di Li, Jiaxiang Lin, Athanasios V Vasilakos, and Muhammad Imran. Security in software-defined networking: Threats and countermeasures. Mobile Networks and Applications, 21:764–776, 2016.

[72] R Shashidhara, Nisha Ahuja, M Lajuvanthi, S Akhila, Ashok Kumar Das, and Joel JPC Rodrigues. Sdn-chain: Privacy-preserving protocol for software defined networks using blockchain. Security and Privacy, 4(6):e178, 2021.

[73] Asad Irfan, Nayab Taj, and Sahibzada Ali Mahmud. A novel secure sdn/lte based architecture for smart grid security. In 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pages 762–769. IEEE, 2015.

[74] Neelam Dayal and Shashank Srivastava. Analyzing behavior of ddos attacks to identify ddos detection features in sdn. In 2017 9th International Conference on Communication Systems and Networks (COMSNETS), pages 274–281. IEEE, 2017.

[75] Prashant Kumar, Meenakshi Tripathi, Ajay Nehra, Mauro Conti, and Chhagan Lal. Safety: Early detection and mitigation of tcp syn flood utilizing entropy in sdn. IEEE Transactions on Network and Service Management, 15(4):1545–1559, 2018.

[76] Justin Clarke-Salt. SQL injection attacks and defense. Elsevier, 2009.

[77] Germán E Rodríguez, Jenny G Torres, Pamela Flores, and Diego E Benavides. Cross-site scripting (xss) attacks and mitigation: A survey. Computer Networks, 166:106960, 2020.

[78] Shakila Zaman, M Shamim Kaiser, Risala Tasin Khan, and Mufti Mahmud. Towards sdn and blockchain based iot countermeasures: a survey. In 2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI), pages 1–6. IEEE, 2020.

[79] Danai Chasaki and Christopher Mansour. Sdn security through system call learning. In 2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pages 1–6, 2021.

[80] Takayuki Sasaki, Adrian Perrig, and Daniele E Asoni. Control-plane isolation and recovery for a secure sdn architecture. In 2016 IEEE NetSoft Conference and Workshops (NetSoft), pages 459–464. IEEE, 2016.

[81] Minh Hieu Nguyen Ba, Jacob Bennett, Michael Gallagher, and Suman Bhunia. A case study of credential stuffing attack: Canva data breach. In 2021 International Conference on Computational Science and Computational Intelligence (CSCI), pages 735–740. IEEE, 2021.

[82] Qiao Yan, F Richard Yu, Qingxiang Gong, and Jianqiang Li. Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. IEEE communications surveys & tutorials, 18(1):602–622, 2015.

[83] Bijeeta Pal, Tal Daniel, Rahul Chatterjee, and Thomas Ristenpart. Beyond credential stuffing: Password similarity models using neural networks. In 2019 IEEE Symposium on Security and Privacy (SP), pages 417–434. IEEE, 2019.

[84] Zhendong Su and Gary Wassermann. The essence of command injection attacks in web applications. Acm Sigplan Notices, 41(1):372–382, 2006.

[85] Anastasios Stasinopoulos, Christoforos Ntantogian, and Christos Xenakis. Commix: Detecting and exploiting command injection flaws. Dept. Digit. Syst., Univ. Piraeus, Piraeus, Greece, White Paper, 2015.

[86] Shuhua Deng, Xing Gao, Zebin Lu, and Xieping Gao. Packet injection attack and its defense in software-defined networks. IEEE Transactions on Information Forensics and Security, 13(3):695–705, 2017.

[87] Fahad M Alotaibi and Vassilios G Vassilakis. Toward an sdn-based web application firewall: Defending against sql injection attacks. Future Internet, 15(5):170, 2023.

[88] Neminath Hubballi, Yogendra Singh, and Dipin Garg. Xssmitigate: Deep packet inspection based xss attack quarantine in software defined networks. In 2023 IEEE International Conference on Consumer Electronics (ICCE), pages 1–6. IEEE, 2023.

[89] Hsing-Chung Chen, Aristophane Nshimiyimana, Cahya Damarjati, and Pi-Hsien Chang. Detection and prevention of cross-site scripting attack with combined approaches. In 2021 International Conference on Electronics, Information, and Communication (ICEIC), pages 1–4. IEEE, 2021.

[90] Dharmendra Choukse, Dimitris N Kanellopoulos, and Umesh Kumar Singh. Developing secure web applications. International Journal of Internet Technology and Secured Transactions, 4(2-3):221–236, 2012.

[91] Raghu Yeluri, Enrique Castro-Leon, Raghu Yeluri, and Enrique Castro-Leon. Network security in the cloud. Building the Infrastructure for Cloud Security: A Solutions view, pages 123–140, 2014.

[92] Jamie Twycross and Matthew M Williamson. Implementing and testing a virus throttle. In 12th USENIX Security Symposium (USENIX Security 03), 2003.

[93] Jianfeng Xu, Liming Wang, and Zhen Xu. An enhanced saturation attack and its mitigation mechanism in software-defined networking. Computer Networks, 169:107092, 2020.

[94] Haopei Wang, Lei Xu, and Guofei Gu. Floodguard: A dos attack prevention extension in software-defined networks. In 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pages 239–250. IEEE, 2015.

[95] Mikhail Belyaev and Svetlana Gaivoronski. Towards load balancing in sdn-networks during ddos-attacks. In 2014 international science and technology conference (modern networking technologies)(MoNeTeC), pages 1–6. IEEE, 2014.

[96] Felix Klaedtke, Ghassan O Karame, Roberto Bifulco, and Heng Cui. Access control for sdn controllers. In Proceedings of the third workshop on Hot topics in software defined networking, pages 219–220, 2014.

[97] Yuchia Tseng, Montida Pattaranantakul, Ruan He, Zonghua Zhang, and Farid Naït-Abdesselam. Controller dac: Securing sdn controller with dynamic access control. In 2017 IEEE International Conference on Communications (ICC), pages 1–6. IEEE, 2017.

[98] Sorin Buzura, Vlad Lazar, Bogdan Iancu, Adrian Peculea, and Vasile Dadarlat. Using software-defined networking technology for delivering software updates to wireless sensor networks. In 2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet), pages 1–6. IEEE, 2021.

[99] Pang-Wei Tsai, Chun-Wei Tsai, Chia-Wei Hsu, and Chu-Sing Yang. Network monitoring in software-defined networking: A review. IEEE Systems Journal, 12(4):3958–3969, 2018.

[100] Vasily Pashkov, Alexander Shalimov, and Ruslan Smeliansky. Controller failover for sdn enterprise networks. In 2014 international science and technology conference (modern networking technologies)(monetec), pages 1–6. IEEE, 2014.

[101] Neerja Mhaskar, Mohammed Alabbad, and Ridha Khedri. A formal approach to network segmentation. Computers & Security, 103:102162, 2021.

[102] Nathan Kong. Design concept for a failover mechanism in distributed sdn controllers. 2017.

[103] D Richard Kuhn, Edward J Coyne, Timothy R Weil, et al. Adding attributes to role-based access control. Computer, 43(6):79–81, 2010.

[104] Harry Wolfson. Out-of-band flow control for reliable multicast. MIT Lincoln Laboratory, 2000.

[105] Eric Rescorla. The transport layer security (tls) protocol version 1.3. Technical report, 2018.

[106] Tatu Ylonen and Chris Lonvick. The secure shell (ssh) protocol architecture. Technical report, 2006.

[107] Steve Vinoski. Advanced message queuing protocol. IEEE Internet Computing, 10(6):87–89, 2006.

[108] Kevin Phemius, Mathieu Bouet, and Jérémie Leguay. Disco: Distributed multi-domain sdn controllers. In 2014 IEEE network operations and management symposium (NOMS), pages 1–4. IEEE, 2014.

[109] Syed Abdullah Shah, Jannet Faiz, Maham Farooq, Aamir Shafi, and Syed Akbar Mehdi. An architectural evaluation of sdn controllers. In 2013 IEEE international conference on communications (ICC), pages 3504–3508. IEEE, 2013.

[110] Jiaqiang Liu, Yong Li, Huandong Wang, Depeng Jin, Li Su, Lieguang Zeng, and Thanos Vasilakos. Leveraging software-defined networking for security policy enforcement. Information Sciences, 327:288–299, 2016.

[111] Amin Tootoonchian and Yashar Ganjali. Hyperflow: A distributed control plane for openflow. In Proceedings of the 2010 internet network management conference on Research on enterprise networking, volume 3, pages 10–5555, 2010.

[112] Andreas Voellmy and Junchang Wang. Scalable software defined network controllers. In Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, pages 289–290, 2012.

[113] Heng Pan, Zhenyu Li, Penghao Zhang, Kave Salamatian, and Gaogang Xie. Misconfiguration checking for sdn: Data structure, theory and algorithms. In 2020 IEEE 28th International Conference on Network Protocols (ICNP), pages 1–11. IEEE, 2020.

[114] Debasish Das, Utpal Sharma, and DK Bhattacharyya. Detection of http flooding attacks in multiple scenarios. In Proceedings of the 2011 international conference on communication, computing & security, pages 517–522, 2011.

[115] Mauro Conti Reza Mohammadi, Chhagan Lal. Httpscout: A machine learning based countermeasure for http flood attacks in sdn. International Journal of Information Security, 22(2):367–379, 2023.

[116] Varun Chauhan and Pranav Saini. Icmp flood attacks: A vulnerability analysis. In Cyber Security: Proceedings of CSI 2015, pages 261–268. Springer, 2018.

[117] Misenga Mumpela Joëlle and Young-Hoon Park. Strategies for detecting and mitigating ddos attacks in sdn: A survey. Journal of Intelligent & Fuzzy Systems, 35(6):5913–5925, 2018.

[118] Bo Hang, Ruimin Hu, and Wei Shi. An enhanced syn cookie defence method for tcp ddos attack. Journal of Networks, 6(8):1206, 2011.

[119] Anchit Bijalwan, Mohammad Wazid, Emmanuel S Pilli, and Ramesh Chandra Joshi. Forensics of random-udp flooding attacks. Journal of Networks, 10(5):287, 2015.

[120] Hung-Chuan Wei, Yung-Hao Tung, and Chia-Mu Yu. Counteracting udp flooding attacks in sdn. In 2016 IEEE NetSoft Conference and Workshops (NetSoft), pages 367–371. IEEE, 2016.

[121] Yung-Hao Tung, Hung-Chuan Wei, Yen-Wu Ti, Yao-Tung Tsou, Neetesh Saxena, and Chia-Mu Yu. Counteracting udp flooding attacks in sdn. Electronics, 9(8):1239, 2020.

[122] Jin Tang, Yu Cheng, and Yong Hao. Detection and prevention of sip flooding attacks in voice over ip networks. In 2012 Proceedings IEEE INFOCOM, pages 1161–1169. IEEE, 2012.

[123] Jonathan Rosenberg and Henning Schulzrinne. Session initiation protocol (sip): locating sip servers. Technical report, 2002.

[124] Henning Schulzrinne, Steven Casner, R Frederick, and Van Jacobson. Rfc3550: Rtp: A transport protocol for real-time applications, 2003.

[125] John Lazzaro. Framing real-time transport protocol (rtp) and rtp control protocol (rtcp) packets over connection-oriented transport. Technical report, 2006.

[126] Mohammad Adnan Aladaileh, Mohammed Anbar, Ahmed J. Hintaw, Iznan H. Hasbullah, Abdullah Ahmed Bahashwan, Taief Alaa Al-Amiedy, and Dyala R. Ibrahim. Effectiveness of an entropy-based approach for detecting low- and high-rate ddos attacks against the sdn controller: Experimental analysis. Applied Sciences, 13(2), 2023.

[127] Mohammad Adnan Aladaileh, Mohammed Anbar, Ahmed J. Hintaw, Iznan H. Hasbullah, Abdullah Ahmed Bahashwan, and Shadi Al-Sarawi. Renyi joint entropy-based dynamic threshold approach to detect ddos attacks against sdn controller with various traffic rates. Applied Sciences, 12(12), 2022.

[128] Tewelde Gebremedhin Gebremeskel, Ketema Adere Gemeda, Gopi Krishna T, and Janaki Ramulu Perumalla. Ddos attack detection and classification using hybrid model for multi-controller sdn, 2022.

[129] Yi Shen, Chunming Wu, Dezhang Kong, and Qiumei Cheng. Flow table saturation attack against dynamic timeout mechanisms in sdn. Applied Sciences, 13(12), 2023.

[130] Ranyelson Neres Carvalho, Jacir Luiz Bordim, and Eduardo Adilio Pelinson Alchieri. Entropy-based dos attack identification in sdn. In 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pages 627–634, 2019.
[131] Jin Wang and Liping Wang. Sdn-defend: A lightweight online attack detection and mitigation system for ddos attacks in sdn. Sensors, 22(21), 2022.
[132] Tianyi Zhang and Yong Wang. Rlfat: A transformer-based relay link forged attack detection mechanism in sdn. Electronics, 12(10), 2023.
[133] Muhammad Reazul Haque, Saw Chin Tan, Zulfadzli Yusoff, Kashif Nisar, Rizaludin Kaspin, Iram Haider, Sana Nisar, JPC Rodrigues, Bhawani Shankar Chowdhry, Muhammad Aslam Uqaili, et al. Unprecedented smart algorithm for uninterrupted sdn services during ddos attack. Computers, Materials & Continua, 70(1), 2022.
[134] Sumit Badotra, Sarvesh Tanwar, Salil Bharany, Ateeq Ur Rehman, Elsayed Tag Eldin, Nivin A. Ghamry, and Muhammad Shafiq. A ddos vulnerability analysis system against distributed sdn controllers in a cloud computing environment. Electronics, 11(19), 2022.
[135] Quan Ren, Zehua Guo, Jiangxing Wu, Tao Hu, Lu Jie, Yuxiang Hu, and Lei He. Sdn-esrc: A secure and resilient control plane for software-defined networks. IEEE Transactions on Network and Service Management, 19(3):2366–2381, 2022.

ZAHEED AHMED BHUIYAN has earned his Master's Degree in Computer Science and Engineering (CSE) from United International University Bangladesh (UIU). Currently, he is working as a Research Assistant under Prof. Dr Md. Motaharul Islam. His major is cybersecurity, and he is interested in the Internet of Things (IoT), Cloud Computing, Cloud Security, Networking, Software Defined Networking, Healthcare Technologies, Green Computing, Artificial Intelligence, Satellite Internet Communication, and Grid-level Energy Storage Systems. He is particularly interested in the Fourth Industrial Revolution (4IR) because of its fusion of AI, robotics, the IoT, and quantum computing advances.

SALEKUL ISLAM (Senior Member, IEEE) received his Ph.D. degree from the Computer Science and Software Engineering Department, Concordia University, in 2008. He is currently a Professor and the Director of the Institutional Quality Assurance Cell (IQAC) of United International University, Bangladesh. Previously, he worked as an FQRNT Postdoctoral Fellow at the Énergie, Matériaux et Télécommunications (EMT) Centre, Institut National de la Recherche Scientifique (INRS), Montréal, Canada. His research interests include future internet architecture, blockchain, edge cloud, software-defined networks, multicast security, security protocol validation, machine learning, and AI. He is serving as an Associate Editor for the IEEE Access and Frontiers in High-Performance Computing journals.

MD. MOTAHARUL ISLAM has been working as a Professor and the director of the Master's Program, Dept. of Computer Science and Engineering at United International University (UIU), Dhaka, Bangladesh. Before joining UIU, Dr. Islam served many other universities at home and abroad, such as the Islamic University of Madinah, KSA, the Islamic University of Technology, Brac University and the University Grants Commission of Bangladesh. He was awarded a Ph.D. in Computer Engineering from Kyung Hee University, South Korea 2013. His research interest includes the Smart Internet of Things, IP-based Wireless Sensor Network (IP-WSN), WSN Virtualization, Cloud Computing, Green Computing, etc. He has published around one hundred articles in the last ten years.

A B M AHASAN ULLAH has earned his Master's Degree in Computer Science and Engineering (CSE) from United International University Bangladesh (UIU). Currently, he is serving as Senior Assistant Vice President, Infrastructure & Network, ICT Security & Risk Dept., ICT Division at LankaBangla Finance Limited, Bangladesh. His major is cybersecurity, and he is interested in the Internet of Things (IoT), Cloud Computing, Cloud Security, Networking, Software Defined Networking. He has a huge interest in the Fourth Industrial Revolution (4IR) because of its fusion of AI, robotics, the IoT, and quantum computing advances.

FARHA NAZ received her bachelor's degree in computer science and engineering (BSCSE) and master's degree in computer science and engineering (MSCSE) from United International University (UIU). She is currently working as a full-time teacher at "Eminent School of Dhaka". Her research includes cloud computing, cloud security, networking, software-defined networking, healthcare hypothesis, application development, tentative application structure, and module design. Her utmost interest is in Software design and management.

MOHAMMAD SHAHRIAR RAHMAN received his B.Sc. degree in computer science and engineering from the University of Dhaka, Bangladesh, in 2006, and the M.S. and Ph.D. degrees in information science from the Japan Advanced Institute of Science and Technology (JAIST), in 2009 and 2012, respectively. He worked as a Research Engineer with the Information Security Group of KDDI Research, Japan. He is currently working as a Professor and also serving as the director of CITS at the United International University, Bangladesh. He has co-authored over 50 research articles and submitted 8 co-authored Japanese patent applications. His research interests include secure protocol construction, privacy-preserving computation, and security modelling. He is a member of the International Association for Cryptologic Research (IACR).