IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 March 2024, accepted 10 April 2024, date of publication 18 April 2024, date of current version 26 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3390968

SURVEY

Exploring Security Dynamics in SDN Controller Architectures: Threat Landscape and Implications

ARUSA KANWAL[1], MOHAMMAD NIZAMUDDIN[2], WASEEM IQBAL[3], WAQAS AMAN[4], YAWAR ABBAS[1], AND SHYNAR MUSSIRALIYEVA[5]

[1]Department of Information Security, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan
[2]Department of Engineering, Physics and Technology, Bronx Community College (BCC), The City University of New York, Bronx, NY 10453, USA
[3]Department of Electrical and Computer Engineering, College of Engineering, Sultan Qaboos University, Muscat 123, Oman
[4]Department of Information Systems, Sultan Qaboos University, Muscat 123, Oman
[5]Information Systems Department, Al-Farabi Kazakh National University, 050040 Almaty, Kazakhstan

Corresponding author: Waseem Iqbal (m.waseem@squ.edu.om)

ABSTRACT Software Defined Networking (SDN) has emerged as a new paradigm for managing heterogeneous networks ranging from enterprises to home network via decoupling the control plane from the data plane. In the traditional networking landscape, these two planes are tightly bound together inside a single appliance. The logically centralized and distributed control plane and programmability offer a great opportunity to improve network security, such as by implementing new mechanisms to detect and mitigate various threats, and also enable security as a service in an SDN paradigm. Due to the ever increasing and fast development of SDN, this paper provides an extensive survey of SDN controllers, SDN-related security threats, and solutions to mitigate the security threats. This study provides a comprehensive survey of 53 SDN controllers from different aspects, including language, architecture, organization, open source, scalability, consistency, reliability, API used, library, and their description. We have also provided a detailed security analysis of SDN architecture with an extensive classification of security threats endangering its different architectural components and the solutions to effectively mitigate them. This paper also identifies challenges and promising future directions on SDN deployment, standardization, implementation, and security issues that should be addressed in this field.

INDEX TERMS Software defined networking (SDN), Openflow, SDN controllers, network operating system (NOS), scalability, SDN attacks.

I. INTRODUCTION

In the current Information Technology (IT) era, communication is based on large networks that are geographically distributed. The distributed approach increases the overall networking complexity [1]. Current network behavior varies from a small number of appliances and sensors to a large number of network devices such as routers and switches. Fixed-function network devices, such as routers and switches, are the foundation of traditional networking. Each of these devices has a specific purpose that works together with the others and helps to sustain the network. The network speed is typically improved if network features are introduced as hardware constructs. A data center is a physical location where the organization keeps its mission critical software and data. A data center's architecture revolves around a network of computing and storage facilities, facilitating the delivery of shared software and data. Routers, switches, firewalls, disk systems, servers, and application delivery controllers are all essential components of a data center architecture. Traditional networking has some drawbacks, such as its static nature, which conflicts with the complex nature of today's server specifications. Because of the complexity of today's network, IT finds it difficult to apply a consistent set of access controls. As a result, traditional policies expose companies to security breaches as well as regulatory or

The associate editor coordinating the review of this manuscript and approving it for publication was Maurizio Casoni.

non-compliance problems. Managing this complexity via the traditional networking paradigm is unpropitious in the rapidly changing networking landscape. As a result, different types of challenges are produced related to networks, such as the management of devices, variations, and security attacks in the network. The networking community came up with a new networking paradigm known as software-defined networks (SDN). SDN is a new technology that decouples the control plane from the data plane, which is tightly bound in traditional networking devices. Mckeown et al. [2] were the first authors to introduce the concept of the OpenFlow protocol. Different IT sectors, such as data centers, academics, government, and enterprises, have promoted the development of SDN. The use of developing technologies like 5G, wireless, the Internet of Things (IoT), and cloud computing can also be improved by introducing SDN-related solutions. There has been a significant increase in the implementation and development of SDN in the current landscape for different types of networks, such as cellular, wide area, IoT, wireless, and data centers.

SDN is a technology that solves many of the traditional networking problems, such as complexity, reliability, security, etc.; however, SDN technology has its own security threat vectors. The attack vector poses a significant concern among the networking industry for its large-scale adaptation.

A. WHY IT IS NEEDED

To the best of our knowledge, we have not seen a comprehensive survey that encompasses all the necessary requirements for the selection of a controller based on its functional and specific features. Some researchers have highlighted the SDN architecture and limited number of controllers only, whereas other researchers have come up with the identification of various security attacks on different planes of the SDN architecture. Therefore, we intend to provide a consolidated overview of SDN architecture, classification of SDN controllers, comparison of controllers and OpenFlow versions, along-with their security attacks and proposed defense mechanisms available in the literature, as shown in Table 1. Finally, we presented some open research issues that will be helpful for academicians and researchers.

B. WHO IT IS INTENDED FOR

This comprehensive survey helps us to address essential questions such as where, why, and which solutions are the most suitable to tackle a specific form of attack on the controller from unlimited mitigation attacks. On one hand, it encourages researchers to choose the most appropriate methodology for the future, while on the other hand, it enables practitioners to take full advantage of this technology and make SDN a promising, trustworthy, efficient, and secure architecture for years to come. The contribution of this research are as follows:

- A comprehensive survey has been carried out on a total of 53 SDN controllers with respect to their programming languages, developers, OpenFlow versions, their architecture, their availability as open source, reliability, scalability, consistency, API, their available libraries, benefits and shortcomings, and the year of launch.
- State-of the art security threats, vulnerabilities, and problems in control, data, and application planes are extensively analyzed in this survey.
- An up-to-date taxonometric classification of different security attacks on SDN architecture.
- A comprehensive overview of these attacks with suggested mitigation controls. At the end of each section, these mitigation strategies are summarized in a table outlining their problems and solutions.
- Identification of research challenges related to SDN to recognize potential future research opportunities.

We have adopted the comprehensive literature review methodology for this study. A comprehensive review uses a literature review method that is specific and thorough to avoid outcome bias. Since it has been shown to be a good tool for summarizing the existing evidence concerning a technology and identifying the gaps, we follow this approach to achieve our goals. We have considered only those papers that provide adequate information about SDN, SDN controllers, different types of security threats related to SDN and some solutions to mitigate the SDN related security threats. This resulted in a series of SDN related papers published between the 2000 to 2020.

The rest of the paper is organized as follows: In Section II, we explain a detailed architecture of SDN. Section III is about the survey methodology, we discussed about the related work in SDN environment, and how our survey is different from the existing ones. In Section IV, we provide a comparison of Openflow versions and a comparison and classification of SDN controllers. We also discuss a variety of security threats related to SDN architecture and those faced by different SDN layers/interfaces. We also suggested potential solutions to mitigate attacks related to the SDN architecture. Research challenges are highlighted in Section V. The paper is concluded in Section VI. The rest of the paper is as followed as depicted in Fig. 1.

II. SDN ARCHITECTURE

SDN architecture is based on four features which are discussed as below.

- Control plane programmability: A lot of efforts have been made for the programmability of networks. One example is the concept of active networking that attempts to control a network in a real-time manner using software. Some of the solutions related to active networking are SwitchWare [6], Bird [7], and Click [8]. In SwitchWare, the network operation can be dynamically modified by allowing a packet to smoothly pass through a network. Network devices can also be programmed by creating software routers, which should be established on PC hardware like Click and Bird. The behavior of the network devices can be changed by using modified versions of different routing software.

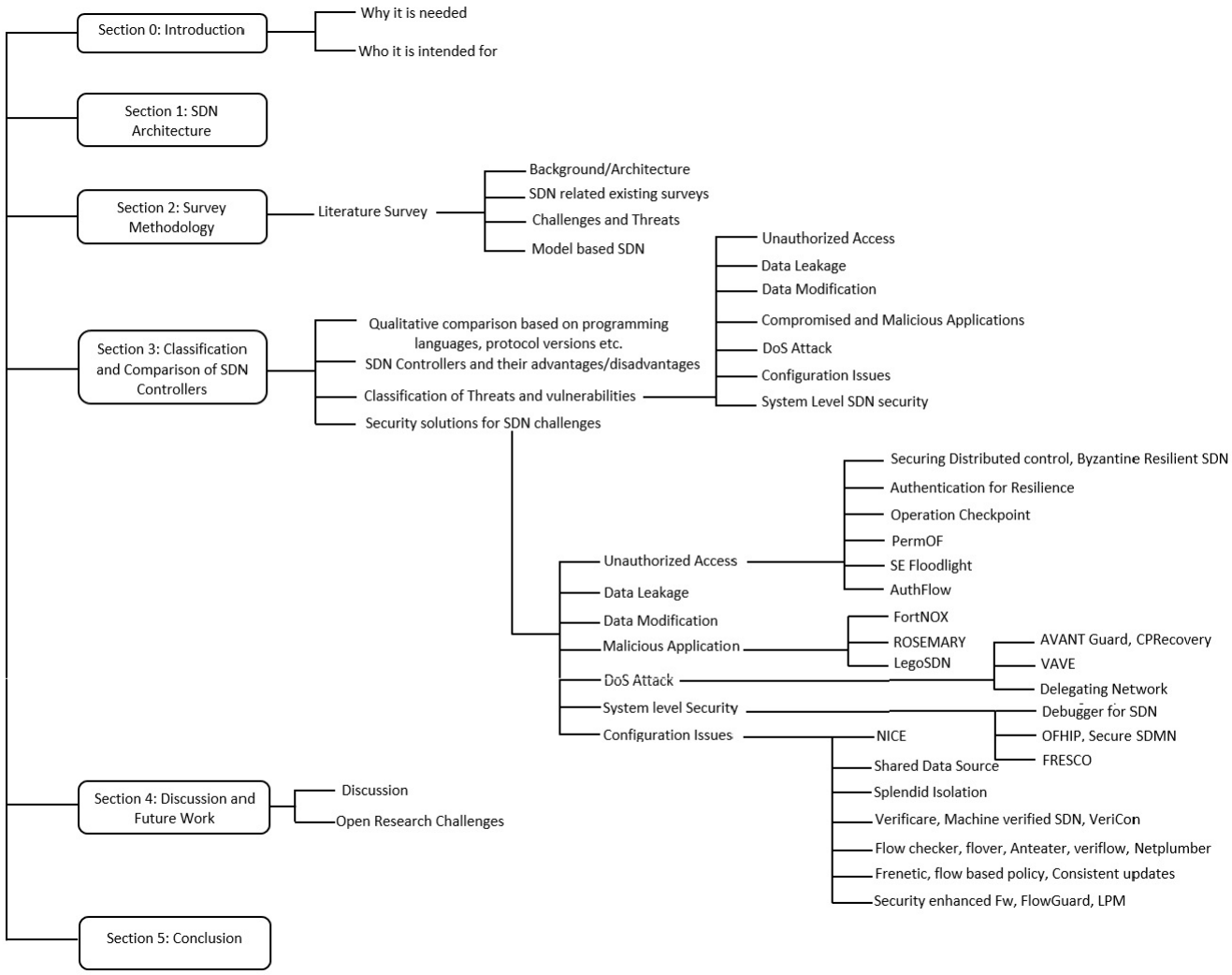

FIGURE 1. Taxonomy of the paper.

- Control plane and data plane decoupling: The spirit has been developed in the last era for separating the data and control planes. In 2004, routing control platform-based research was proposed by Caesar et al. [9], in which the path computation complexity can be reduced by using centralized routing instead of the border gateway protocol (BGP).
- Forwarding decisions can be based on flow instead of destination. A flow is broadly defined by a set of packet field values acting as a match criterion and a set of actions. In SDN, the packet sequence between the destination and the source is defined as the flow. Identical service policies can be received at forwarding devices during the flow of packet [157].
- Control logic is rearranged as a new entity with the name of Network Operating System (NOS) or SDN controller. The purpose of using the NOS and the traditional operating system is quite similar, as the NOS runs on commodity server technology, reducing both capital and operational costs.

This section presents a comprehensive overview of the SDN architecture that we decomposed into three main layers, as depicted in Fig. 2; data, control, and application plane. The main components of SDN architecture are explained as follows:

A. SOUTHBOUND INTERFACE

Southbound interface behaves as a communication protocol between the control and the data plane. The interaction between the control and data planes can be formalized by using this protocol. The southbound API is part of the southbound interface and defines the flow rules related to networking devices.

B. NETWORK DEVICES

Network devices perform packet forwarding. These devices have defined rules through which the in-coming packet can be either forwarded, dropped, or rerouted to the destination. The southbound interface defines these rules, and the controllers

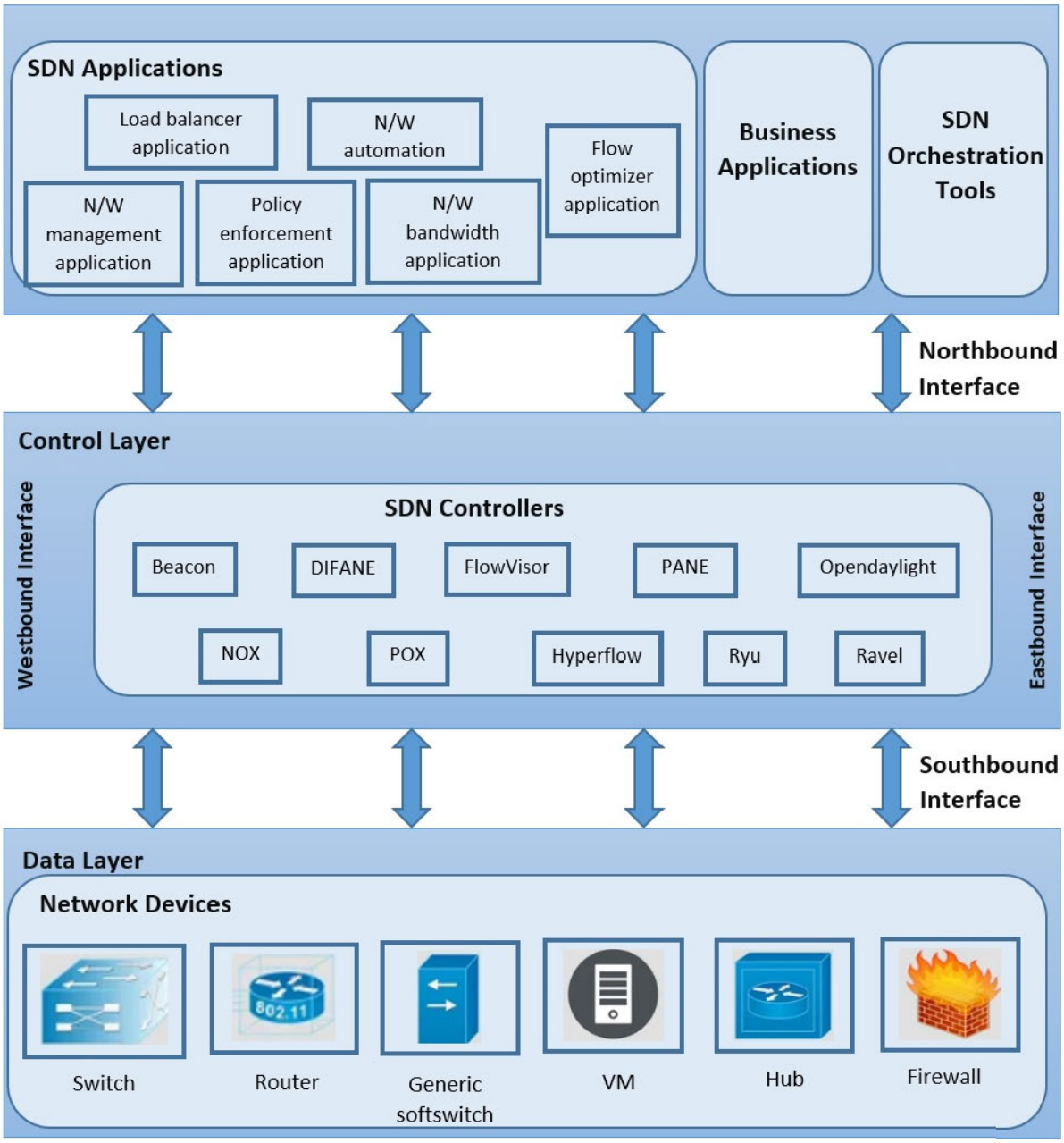

FIGURE 2. SDN architecture with its main layers and interfaces: data plane layer, control plane layer, application/management plane layer, southbound interface, northbound interface, eastbound interface and westbound interface. Infrastructure layer handles the data processing, control layer governs the rule, and application layer encompasses different business/industrial applications.

install them on forwarding devices by using southbound protocols.

C. DATA AGENT

SDN southbound interface defines the number of data agents that might be used in the networking devices for SDN controller interaction and parules forwarding. ForCES and the OpenFlow agents are some examples.

D. DATA PLANE

The data plane of SDN works similar to the physical layer of an OSI model, and it is the bottom layer of the architecture, consisting of the physical devices that deal with the data traffic. This plane consists of network devices, such as firewalls, switches, and IDSs, through which packets can be defined and forwarded. These network devices are interconnected through wired or wireless connections.

E. SDN CONTROLLER

The SDN controller acts as the brain of the SDN architecture. Core management functions like synchronization and management of the devices are provided by the controller. By using the controllers or northbound API, installation of applications can be enabled in the data plane.

F. CONTROL PLANE

The control plane is an important part of SDN where the controllers are located. Rules and functionalities for controlling the network can be implemented by using the southbound interface through the data plane. A control plane is used to install or configure some control information. The control plane does not have all the traffic. Controllers install or change the flow table, either actively or reactively, upon the occurrence of some events. When network operation starts, the controller does not load the flow table with any rules in a reactive approach. The controller can install rules into the flow table as packets arrive at switches during the network operation [158]. The controller can add flow entries to the flow table in advance for the proactive approach. The choice of rules is critical for maximizing network efficiency, especially in large-scale networks like data centers [146]. Management and configuration of controllers are major functions of the control plane. Instructions can be received from the application layer and then passed into the network. Routing protocols such as RIP, OSPF, and BGP are managed by the control plane with the help of IpV4 & 6.

G. EASTBOUND AND WESTBOUND INTERFACE

It is important to implement the eastbound and westbound interfaces as the control plane is physically distributed. The connections between the distributed controllers can be enabled by using these interfaces.

H. NORTHBOUND INTERFACE

The Northbound API lies between the application and the control layers. Applications can be developed by using the SDN controllers and NOS. Communications between the users and the forwarding devices can be done through this interface. The northbound interface can be implemented by using the common interface of controllers.

I. APPLICATION PLANE

Network applications are included in the application plane. Different types of application are deployed through which network management operations can be implemented to control the functionalities of the applications. Policies can be defined for the instructions for performing different types of tasks.

J. APPLICATIONS

Network control logic can be defined by using software programs. Applications are used in application plane to access the network through controller API or through northbound interface respectively. Application modules, or control programs, are other terms used for controller-based applications.

III. SURVEY METHODOLOGY

In this section, we describe the main procedures we performed to conduct this review. We adopted the methodology of comprehensive literature review. A comprehensive review uses literature review method that are specific and thorough to avoid the outcome bias. We consider only those articles that provide adequate information about the SDN, SDN controllers, different types of security threats related to SDN, and some solutions to mitigate the SDN related security threats. The search was conducted for articles published between the years 2000 and 2023.

A. LITERATURE SURVEY

Current literature provides a spectrum of architectures, challenges, threats, mitigation controls, and models related to SDN. Below we present a top-level review of such related works identified and we also grouped the survey papers.

1) BACKGROUND/ARCHITECTURE

Abbasi et al. [14], provided a detailed description of software defined cloud computing architecture. They also discussed the implementation challenges, such as scalability, programmability, security and interpretability related to software defined cloud computing and the possible solutions of the challenges.

A detailed description about the background and structure of SDN was presented by Zhang et al. [24]. They discussed about the layers and the technologies related to the layers. The authors also compares the pros and cons related to those technologies of each layer. At the end of their research, they also discussed about open research challenges related to SDN.

Xia et al. [25] surveyed about the concept and the benefits of using SDN. The authors provided details by which performance, innovation and configuration can be improved for the SDN. The authors also discussed the current research in the three layers of SDN.

A detailed description about the architecture of Openflow and SDN is presented in [28]. The authors Jammal et al. also discussed about the challenges of SDN and their existing solutions to solve the challenges. Some of the SDN challenges were based on reliability, performance, security, low level interface, controller placement, scalability and CPU limitations. At the end, the authors also discussed latest SDN tools and their implementations.

Scott-Hayward et al., presented information about the architecture of SDN in [32]. Their survey was based on both the areas of industry and research community. They also discussed about the possible attacks related to SDN and their possible solutions to protect the system. They also discussed on methods for improving the network security by using the SDN architecture characteristics.

Li et al. provided a detailed description related to background of SDN and Openflow in [33]. They discussed the different security challenges related to Openflow-based SDN. These challenges were related to controllers, switches, channels based on confidentiality, integrity and availability model. They also explain the existing solutions related to the categorized challenges.

The concept of SDN and its implementation was discussed by Ahmad et al. [35]. They stressed that scalability and the robust security can be improved by combining the network programmability and the control plane. They also reviewed benefits and the security challenges related to application, control and data planes. Numerous security solutions of the challenges related to different planes of SDN were also underlined. Furthermore, the authors discussed the techniques that could improve network-wide security of SDN. Finally, the authors provided a solution based on the recommendation of International Telecommunication Union.

Feghali et al., discussed the architectural components of SDN in [37]. They also discussed the communication protocols, such as Openflow and Opflex. Moreover, the authors also presented the security issues concerning SDN communication, component, logging, and audit level. Issues related to northbound and southbound communication were discussed at communication level and those related to switch, controller and application were discussed at component level. At the end of their survey, the authors discussed some existing solutions and their limitations to the issues.

In [43], If, Akyildiz et al. provided an overview of traffic engineering and SDN. Their main focus was on four attacks: fault tolerance, updates related to topology, flow management and traffic characterization. The authors also discussed the tools of traffic engineering that were used in both academic and industrial settings. The authors also discussed about the important aspects of scalability, consistency, and availability in perspective of SDN data networking. At the end, they explained the methods of traffic engineering for SDN networks.

History and programmability of SDN networks was discussed by BA. Nunes et al. in [44]. They explained the architecture of SDN and the Openflow standards. They also discussed the testing and implementation alternatives for the SDN-based services and protocols. They provided the details about the SDN applications, the advantages, and the research challenges.

History of SDN and the specification related to Openflow discussed by Lara et al. [45]. They discussed different versions of Openflow that existed at that time. The authors also provided the details about the different applications related to the presented versions. They also explained the advantages and the challenges of the SDN-based networks. At the end, they explained and compared the different specifications of Openflow networks.

2) SDN RELATED EXISTING SURVEYS

DS. Rana et al. [12], surveyed about the development of SDN and latest software tools used in SDN, such as ONIX, Mininet, McNettle and Veriflow. The authors also discussed about the challenges related to SDN such as reliability, security, scalability, performance, and low-level interfaces.

In [17], Zhu et al. performed a qualitative comparison for different SDN controllers and also conducted the quantitative analysis and bench-marking on SDN controllers. The authors also provided a comprehensive survey on bench-marking tools that were used for evaluating the performance of SDN controllers.

Amin et al. [21], provided a comprehensive survey on hybrid SDN network. The authors categorized their survey and discussed about the network deployment, network management techniques, controllers, security and testing for hybrid SDN networks.

Evaluation tests for five controllers was performed by Abdullah et al., in [22]. They also analyzed their performance. Their comparison was based on end-to-end throughput and delay. They used different switches in Mininet emulator to evaluate the performance of each controller to achieve the results. Their purpose of using different switches is based on the fact that the throughput and number of switches are inversely proportional to each other; if one increases then the other decreases. Finally, their result confirmed that ONOS and libfluid controllers had lowest and highest delay respectively as compare to other controllers.

A survey on the challenges related to Internet of Things (IoT) was performed by Wang and Wu [26]. The authors discussed about the benefits and shortcomings of IoT. They also provided the solutions based on fog computing, NFV, cloud computing, and SDN to solve the mentioned challenges.

Mekki et al. [153] presents a comprehensive survey of Software-Defined Networking (SDN) techniques applied to vehicular networks. The paper major goal was to investigate the potential benefits, problems, and possibilities provided by SDN in the context of automotive networks. As a revolutionary networking architecture, SDN decouples the network's control plane from the data plane, allowing for greater flexibility and adaptability. The authors begin with an overview of vehicular networks, describing their distinct traits and problems, such as high mobility, intermittent connectivity, and varied network topologies. They then look into the fundamentals of SDN, explaining the architectural components and essential concepts that distinguish it from traditional network systems. The authors give case studies, simulations, and experimental data from various projects and initiatives in the field throughout the survey, demonstrating the feasibility and usefulness of SDN-based solutions for automotive networks.

Maleh et al. [154] presented a detailed examination of the security aspects associated with Software-Defined Networking (SDN). The paper begins with an overview of SDN, emphasising its basic design, which separates the control plane from the data plane. While SDN provides network management flexibility and agility, it also brings additional security risks that traditional networks did not have. The authors talk about the numerous security threats that can occur in SDN setups. Among these dangers include Distributed Denial of Service (DDoS) assaults, unauthorised access, data breaches, and flow table manipulation. The authors emphasize the significance of these risks and their possible influence on network operations through a

thorough examination. To address the vulnerabilities in SDN systems, the paper goes on to list a variety of security measures and mitigation. These include procedures for access control and authentication, encryption, flow monitoring, anomaly detection, and intrusion prevention systems. The authors thoroughly examine each approach's effectiveness in mitigating specific threats, providing significant insights for network administrators and security practitioners. This research is a helpful resource for researchers, network practitioners, and policymakers in their attempts to construct secure and resilient SDN infrastructures since it provides a thorough examination of threats, countermeasures, and future prospects.

Yeganeh and Ganjali [27] focused on the provision of a comprehensive survey on SDN controllers, which is a sub-category of distributed SDN. The authors also discussed about the architecture and the concept of distributed SDN as compare to traditional SDN. They also provided the logical and practical classification of SDN. A detailed survey on SDN was presented by Kreutz et al. [40]; they discussed about the history of SDN and their standardization activities. They also discussed in detail about the different SDN layers: Infrastructure, southbound and northbound interfaces, NOS, network hyper-visors, programming languages, network applications and virtualization. In addition, the authors discussed some recent research and the issues related to controllers, resilience, security, scalability, migration and performance evaluations in SDN network. A detailed analysis for different SDN controllers was presented by Shalimov et al. [46]. The authors performed some efficiency indexes experiments on the controllers and these indexes were related to security, reliability, scalability and performance. They developed their own framework with the title of 'hcprobe' for doing experiments on controllers. The experimental results show that the performance of the controllers has to be improved in order to work efficiently in above mentioned indexes.

3) CHALLENGES AND THREATS

In [10], Varadharajan et al. proposed a policy-based security architecture for SDN, in which they focused on secure communication between inter and intra domain and detection and prevention of different types of security attacks in the networks. The benefit of using this approach was to provide defense against data flow related attacks which can secure data plane path and end hosts.

In [11], Sultana et al. main focus was to detect network intrusion in SDN by using machine learning and deep learning techniques. The advantage of using their technique was its efficiency that was used for data taxation and for evaluation of the network security. The shortcoming of the paper is its need for an extensive dataset based on the deep learning technique for the evaluation of the performance of the methodology.

Alsaeedi et al. [13] provided an overview and the challenges related to SDN flow control based on network scalability and performance. In addition, they also addressed the challenges by improving the scalability and performance of SDN flow control. The authors summarized the challenges that required to be further addressed for obtaining a scalable and adaptive SDN open flow control.

In [16], a detailed survey presented by Yu et al. on vulnerabilities of SDN and their management solutions for both industrial and academic research. They compared and analyzed the detailed solution on the fault tolerance, fault recovery, fault diagnosis, and monitoring. They also performed the analysis of solutions that were developed either in industry or in academic research context. The authors also suggested some future challenges and emerging trends in the advancement of SDN.

Wenjuan et al. [152] proposed a novel approach to enhance the security of Software Defined Networking (SDN) through the integration of blockchain technology with collaborative intrusion detection mechanisms. SDN is a network architecture that allows for centralized control and programmability of the network, but it faces significant security challenges due to its centralized nature. By harnessing the advantages of blockchain technology, the proposed system exhibits enhanced resilience, data integrity, and cooperative defense mechanisms, making it a valuable contribution to the field of network security in SDN environments.

In [18], Raghav and Dua discussed about the different types of threats and a few security flaws at different SDN layers. This paper mainly focused on the solution provided by the authors to one of the threats. Their solution was based on set intersection and did not take much time for conflict detection.

Raghunath and Krishnan [20] discussed different scenarios of cross-layered attack on SDN architecture. The authors also provided a detailed overview of protecting SDN-enabled network. The authors also provided a comprehensive survey on different threats related to SDN and their defensive mechanism to deal with those threats.

Alsmadi et al. surveyed about the security of SDN in [29]; they discussed about the issues and problems related to SDN. They also provided possible solutions to counter the problems. Their survey was about the security threats and their countermeasures, based on three categories: network intrusion, application trust management, and DDoS attacks. Based on the issues and countermeasures, the authors also provided some future directions for SDN security.

Background of SDN and some security challenges related to SDN was discussed by Bouras et al. [31]. They also discussed about some existing solutions to mitigate the challenges. They performed their experiment by attacking the mobile network an ONOS controller.

Slavov et al. [34] discussed about the benefits and the different types of vulnerabilities related to SDN. The authors also discussed the objectives, such as confidentiality, integrity and availability, which could be protected by keeping the process of authentication, authorization, repudiation,

TABLE 1. Comparison of some existing surveys related to SDN focusing on different parameters inside SDN paradigm.

Existing surveys	Consolidated introduction of SDN	Architecture of SDN	Classification of SDN controllers	Comparison of SDN controllers	Comparison of Openflow versions	SDN related security attacks	Security attacks on different layers of SDN	Defense against the SDN security attacks	Open Research Issues
[10]	Generalized introduction of SDN	×	×	×	×	×	×	×	×
[12]	×	Provide Summary of SDN architecture	×	×	×	×	×	×	×
[13]	×	×	×	Comparison of distributed based SDN Controllers	×	×	×	×	×
[14]	×	Summary of SDN Architecture	×	Comparison of some famous SDN Controllers	×	×	×	×	×
[17]	×	Layeblack structure of SDN	×	Did the comparison for less numbers of controllers	×	×	×	×	×
[20]	×	×	×	×	×	×	×	Defend against the DoS Attack only	×
[22]	Generalized Introduction of SDN	×	×	Comparison of five SDN Controllers	×	×	×	×	×
[24]	×	Summary of SDN Architecture	×	×	✓	×	×	×	×
[25]	Detailed introduction of SDN	×	×	×	✓	×	✓	×	×
[29]	×	Detailed description of SDN Architecture	×	Comparison of some SDN Controllers	×	×	✓	×	×
[30]	Introduction of SDN	×	×	×	×	Cloud based SDN Challenges	×	×	×
[34]	✓	Detailed description of SDN Architecture	×	×	×	Some of the SDN Security Issues	×	Solutions for some of the SDN Security Issues	×
[35]	Detailed introduction of SDN	×	×	×	×	Security challenges related to SDN	×	×	×
[39]	✓	SDN Architecture	×	×	×	SDN Security Issues	×	×	×
[46]	Introduction of SDN	×	×	×	Comparison of some SDN Controllers	×	×	×	✓

transparency, resiliency and multi-domain isolation in a right place.

In [39], Chourishi et al. proposed the three controllers architectures which was implemented for OpenStack, Openflow, and OpenDayLight. The authors also discussed the issues related to security of SDN and load balancing. Their contribution was a major milestone in the IT community by load balancing and network architecture. Architectures of the SDN and the challenges related to the architecture were also discussed in [42]. Some of the principles and the security requirements of the SDN were also presented in this paper. After performing the experiments, they suggested that the principles can guarantee a safe implementation of ONF based architectures. In addition, they presented the analysis on Openflow version 1.3.5.

In [151], Logeswari et al. main objective was to design and implement an efficient Intrusion Detection System (IDS) using machine learning techniques, capable of detecting and mitigating network intrusions in SDNs. They introduces a promising approach to tackle security challenges in SDNs. The integration of multiple machine learning algorithms allows the system to adapt dynamically and respond effectively to emerging threats.

In [155], Akhunzada et al. intends to provide an in-depth review of the security challenges faced by Software Defined Networks. The authors begin by emphasizing the increasing popularity of SDN and the fundamental shift in network management it represents. They underline SDN's sensitivity to numerous security risks and their possible impact on network infrastructure. The study digs into the taxonomy of security concerns unique to SDN, explaining the many types and associated hazards. Furthermore, the authors explain the critical needs for protecting SDN and identify outstanding concerns that must be addressed. Overall, the introduction sets the stage for a thorough examination of SDN security and serves as the foundation for the remaining portions of the paper.

In [156], Akhunzada et al. addresses security and dependability challenges in software-defined networks. The authors examine the weaknesses and potential threats in SDNs and present a comprehensive architecture for ensuring security and dependability in these networks. They explore the need for adding security measures at all tiers of the SDN architecture and show several ideas and techniques for increasing SDN resilience to assaults. The report also emphasizes the importance of secure communication and suggests a secure communication paradigm for SDNs. Furthermore, the authors thoroughly evaluate their suggested framework using simulations and real-world experiments to show its effectiveness in minimizing potential security threats.

4) MODEL BASED SDN

A security framework titled as StateFit was presented by RH. Hwang et al. [15], in which they filtered the traffic at the SDN data plane. Latency and signaling overhead can be reduced by using this framework.

In [19], Abdulqadder et al. proposed a multi-level security mechanism for SDN architecture. Different types of methods were used to examine the packet flow and get the controller for further processing. The main aim was to provide a flexible packet forwarding mechanism in SDN. In the first step, users were verified through routers, and policy verification was done in the next step by using the logic design. Finally, authentication of controllers were done before the flow control transmission by using the signature-based authentication. Their main focus was to improve the security of SDN environments.

A new framework was proposed by Wang et al. [26] for SDN environment by joining the SDN technologies and some security tools with the title of 'SecControl'. The authors also evaluated the performance of the framework 'SecControl' by implementing the prototype with an Openflow protocol. The result of their research confirmed that the framework has worked well with security tools and have defensive response for SDN networks.

In [41], Lantz et al. described a new prototype for SDN in which they deployed a network by using Mininet. The results from the case studies showed that the Mininet is efficient in perspective of resources and usage of time as compared to other workflows. Combining the SDN with the Mininet performed well in the three phases: deployment, sharing and prototyping.

A Cloud-based SDN was discussed by Yan et al. [38]. They provided the details about the new characteristics of DDoS attacks that were related to the SDN-based cloud computing. The authors also presented a detailed survey of all the defense mechanisms against the attacks related to cloud computing. The solutions against those attacks were classified into three categories: source-based, network-based, and the destination-based mechanisms. At the end, the authors discussed the methods of mitigation of mobile and application level DDoS attacks.

The attack surface of SDN was examined by Yoon et al. [150] using an analysis model based on the CIA (confidentiality, integrity and availability) triad. They divide the assets of the SDN architecture into three categories; control plane where they create a threat scheme that can target both the controllers and the application, control channel and data plane. The authors include a list of potential vulnerabilities for each attack. At the same time, the authors reveal 12 previously unreported vulnerabilities. They have found that the newly discovered vulnerabilities mostly fall into two categories; controller intrinsic resilience and access control. The results of this survey demonstrate the need for improved controller stability in order to secure both its own sensitive resources and trustworthy applications by using the strict authentication and authorization mechanisms.

IV. CLASSIFICATION AND COMPARISON OF SDN CONTROLLERS

In the following subsection IV-A, we conducted a qualitative comparison among different SDN controllers based on different features. Description of different SDN controllers also discussed in subsection IV-B.

A. QUALITATIVE COMPARISON BASED ON PROGRAMMING LANGUAGES, PROTOCOL VERSION ETC

Comparison among different SDN controllers can be analyzed based on different features which are mentioned as below.

1) PLATFORM (PROGRAMMING LANGUAGE)

Different types of programming languages can be used for writing the controllers operating system. Languages can be C, C++, C#, Java, R, Python, Ruby, and Go. Some of the controllers use one programming language to write the complete controller. However, other controllers use different languages for different modules of the controller. Therefore, it is a possibility that controllers which are developed with different programming languages can have a better performance on different platforms under special conditions.

2) DEVELOPERS

Different developers like Nicira, Google, Barkeley, Stanford, Juniper, Big Switch, Cisco, NTT and NEC etc., have developed multiple SDN controller.

3) OPENFLOW VERSION

Different SDN controllers deploy the different versions of Openflow. In March 2008, the first version of the Openflow was released with the title of v 0.2.0. After two month, in May 2008, two new versions of the Openflow, v 0.8.0 and v 0.8.1, were released. In the subsequent version of Openflow in Oct 2008, echo request and the reply message methodology were introduced in v 0.8.2. Few updates were introduced in the next version of Openflow v 0.8.9 in December 2008. In July 2009, a new version of Openflow v 0.9.0 was released and in December 2009 the most deployed v 1.0 was released. After that, a new version of Openflow v 1.0.0 was released with some new specification. Payload Ethernet and the 12 header fields were used by different switches to support the specifications of version 1.0.0. A new version 1.1.0 was released, as the v 1.0.0 has only one flow table in a switch but in the new version numerous group and the flow table were presented in a switch.

In December 2011, v 1.2.0 was released with some additional features such as option of multiple controllers been connected to a single switch and the Ipv6 addressing were included. In June 2012, another version 1.3.0 of Openflow was introduced. Connections between the controller and the switch were enabled. Cookies were also added to those packets which came from switch to the controllers. In August 2013, v 1.4.0 of Openflow was released which increased the scalability. In December 2014, v 1.5.0 was

released in which the Ingress and Egress tables were introduced instead of using the input port. Comparison of the specifications of Openflow versions 1.0.0, 1.1.0, 1.2.0, 1.3.0, 1.4.0, 1.5.0, 1.5.1 and 1.6.0 is shown in Table. 2.

4) ARCHITECTURE

Architecture plays an important role in the SDN paradigm as the architecture is characterized into two basic types: distributed and centralized. Distributed and the centralized controllers can be used for the large and the small area network respectively. It can be further classified into flat and hierarchical, in which the root controllers are added in the hierarchical and equal responsibilities are assigned for each controller in the flat category. The classification of SDN controllers based on architecture is shown in Fig. 3.

5) OPEN SOURCE

open source refers to a source code of any program that is available for use or modification by users or other developers. SDN controllers may be differentiated on open source feature as well.

6) RELIABILITY

Reliability is the degree to which an assessment tool produces stable and consistent results. Reliability is an important factor of SDN controllers. The decoupling features of the SDN technology give us the flexibility of independent management and development of control plane and data plane. In addition, two approaches are used for SDN deployment: centralized and de-centralized [147]. In centralized approach, we have a single point of failure [148]. To tackle this issue, multiple SDN controllers are used to provide reliability [148]. However, the de-centralized come up with its own issues.i.e. consistency.

7) SCALABILITY

Major challenges related to scalability are the control logic concentration in software controllers and the separation of data and control planes. Centralized SDN controllers are high in demand when the size of network is increased, and, as a result, problem of data overloading can occur. Communication between the switches and the controllers can be increased when the size and operating region of the network are expanded. A single controller may be the bottleneck in a centralised software defined environment. Consistency and scalability are the major challenges with this approach. With a response time of less than 10 milliseconds, a single NOX SDN controller can handle 30000 flow initiation requests per second [75]. The message processing latency becomes a non-negligible factor in overall network scalability whenever a load on a single controller exceeds its threshold value [149].

8) CONSISTENCY

Consistency is an important factor of distributed SDN controllers. Network state consistency and the network information can be exchanged in physically distributed SDN controllers.

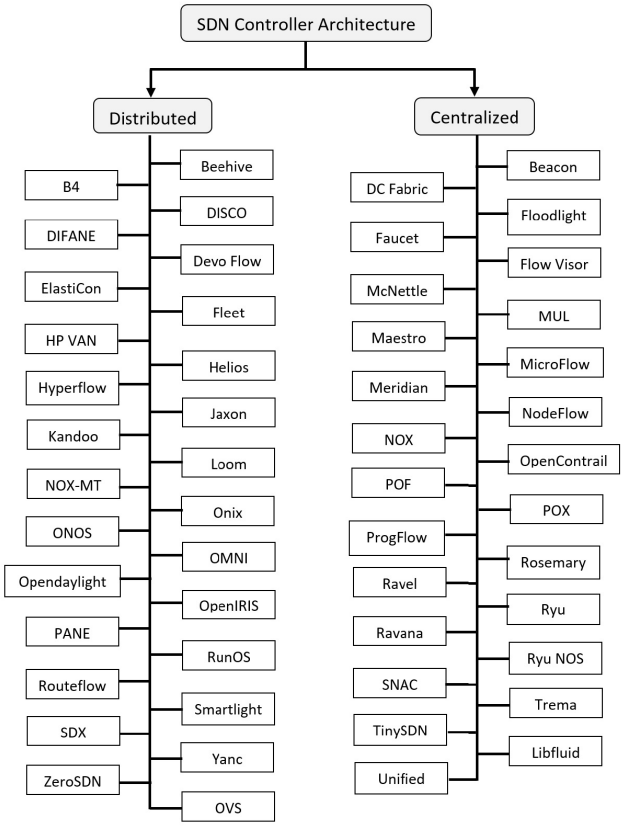

FIGURE 3. Classification of SDN control plane architecture; distributed vs centralized.

9) COMPILER

The compatibility of SDN controllers with OS are defined by the compilers. Some of the major OS are; Linux, Windows, Ubuntu and MacOS. Linux is one of the most used OS for SDN controllers.

10) API

Applications like the virtualization, forwarding, and intrusion detection were created by data planes to facilitate the controllers in northbound API. Communication between the routers, switches, and the controllers can be done by using the southbound API. Multiple controllers from different domains are used to peer with each other by using the west or east API.

11) LIBRARY

Different types of Openflow libraries are used for different SDN controllers. Types of different libraries are depicted in Table 3.

B. SDN CONTROLLERS AND THEIR ADVANTAGES/DISADVANTAGES

The following paragraph highlight different features of the available controllers till date. Following parameters like platform, developers, architecture, reliability, scalability,

TABLE 2. Comparison of openflow versions; 1.0.0, 1.1.0, 1.2.0, 1.3.0, 1.4.0, 1.5.0, 1.5.1 and 1.6.0.

Open flow Version / Specifications	V 1.0.0	V 1.1.0	V 1.2.0	V1.3.0	V1.4.0	V1.5.0	V 1.5.1	V 1.6.0
Year	2009	2011	2011	2012	2013	2015	2015	2016
Widely Organized	×	✓	✓	✓	✓	✓	✓	✓
Flow Table	Single flow table	Multiple flow table	Multiple flow table	Multiple flow table	Multiple flow table	Multiple flow table	Multiple flow table	Multiple flow table
MPLS Matching	✓	×	×	×	×	×	×	×
TCP Flag Matching	✓	✓	✓	✓	✓	×	×	×
Group Table	✓	×	×	×	×	×	×	×
IPv6 Support	✓	✓	×	×	×	×	×	×
Simultaneous communication with multiple controllers	✓	✓	×	×	×	×	×	×
Egress Table	✓	✓	✓	✓	✓	×	×	×
Flow Monitoring	✓	✓	✓	✓	×	×	×	×

consistency, compiler, API and library are the prime focus surveyed in this section in order to give the readers a complete picture of the controllers for deployment according to their scenario/need. A summary of different controllers based on different features is shown in Table 3.

1) BEEHIVE [85]

Distributed architecture-based open source SDN controller established in 2016 by NERSC, operates with Openflow versions 1.0.0 - 1.2.0. It uses Go as programming language, REST API and Linux OS as a compiler for evaluating the controller. Beehive SDN controller provide a better scalability and reliability as it does not have consistency related challenges. Beehive SDN controller is significantly effective in designing the control plane, making this as its main advantage.

2) BEACON [63]

In 2010, an open source SDN controller based on centralized architecture established by Stanford University, called Beacon. Beacon operates with the Openflow version 1.0.0 and have use the Java programming language. It is not able to achieve the goals of scalability and consistency however, it performs well in perspective of reliability. JSpring library, ad-hoc API and Linux, Windows, MacOS use as compiler to evaluate the performance of the controller.

3) B4 [5]

In 2013, Google established a physically distributed architecture-based open source SDN controller. Linux OS uses as a compiler for evaluating the controller performance. B4 operates with the Openflow version 1.0.0 and have use the C and Python as programming language. Its main advantage is the maximization of the average bandwidth. B4 SDN controller provides greater reliability and scalability however, the performance is limited in achieving the goal of consistency.

4) DISCO [64]

DISCO, established by Stanford University in 2014, is based on physically distributed architecture and is an open source SDN controller that operates with Openflow version 1.0.0. The result of the scalability and reliability are achieved in DISCO SDN controller. Java is use as a programming language, JVM library, REST API and Linux OS are used as a compiler for evaluating the performance of controller. The benefit of DISCO controller is the usage of lightweight intercontroller channel, which enabled end-to-end services.

5) DCFABRIC [65]

OpenStack Juno established an open source SDN controller based on Centralized architecture in 2014, which operates with the Openflow version 1.3.0 using the REST API. The desirable results of reliability, scalability and consistency can be attained by DCFabric controller. DCFabric provides high efficiency in network topology. This is the main benefit as they use the JavaScript and C as programming language and Linux OS as compiler for controller evaluation performance.

6) DIFANE [66]

An open source SDN controller, 'DIFANE', based on physically distributed architecture was established in 2010. It operates with the Openflow version 1.1.0 and use the Adhoc API for controller evaluation. DIFANE's performance is limited in achieving the results of consistency and reliability, but it attains the results of better scalability.

7) DEVOFLOW [67]

NEC, Arista, and Toroki, in 2010, established an open source SDN controller 'DevoFlow'. It was based on physically distributed architecture. It uses the JAVA as programming language, JVM library for controller evaluation performance and it operates with the Openflow version 1.1.0. The main advantage of DevoFlow is that it reduce the chance of overhead, while the performance is limited in reliability and consistency. However, it attains the desirable results for scalability.

8) ELASTICON [68]

In 2014, NSF and IIS established a distributed architecture-based open source SDN controller 'ElastiCon', which operates with the Openflow version 1.0.0 and use JAVA as a

TABLE 3. A detailed summary of SDN controllers-highlighting platform, developers and detailed architecture.

SDN Controller	Platform (Language)	Developers (Organization)	Open Flow Version	Architecture	Open Source	Reliability	Scalability	Consistency	Compiler	API	Library	Year
NOX	C++, Python	Nicira, Stanford	V 1.0.0	Centralized	✓	✓	×	×	Linux	Ad-hoc	-	2008
FlowVisor	C	Stanford, Nicira	V 1.0.0 – 1.3.0	Centralized	✓	×	✓	×	Linux	JSON RPC	-	2009
Maestro	Java	Rice University	V 1.0.0	Centralized	✓	✓	✓	×	Linux, MacOS and Window	Ad-hoc	JVM	2010
HyperFlow	C++	U Toronto	V 1.0.0	Physically distributed logically centralized	✓	✓	✓	×	-	-	Boost serialization	2010
DevoFlow	Java	NEC, Arista and Toroki	V 1.1.0	Physically Distributed Logically Centralized	✓	×	✓	×	-	-	JVM	2010
DIFANE	-	-	V 1.1.0	Physically Distributed Logically Centralized	✓	×	✓	×	-	Ad-hoc	-	2010
Beacon	Java	Stanford University	V 1.0.0	Centralized multi-threaded	✓	✓	×	×	Linux, MacOS and window	Ad-hoc API	Open flow JSpring	2010
Onix	C++, Python and Java	Nicira, NTT and Google	V 1.1.0	Physically distributed logically centralized	×	✓	✓	×	-	Onix API, NVP NBAPI	Third part lib	2010
Programm eableFlow	Java	NEC	V 1.0.0	Physically Centralized	✓	✓	×	×	Linux	Control API	-	2010
NOX-MT	C++	Big Switch, Nicira and U Toronto	V 1.0.0	Physically distributed	✓	✓	×	✓	Linux	Ad-hoc API	ASIO lib	2011
OMNI	Python and Java	Federal university of Rio de Janerio	V 1.1.0	Distributed	✓	×	×	×	Linux	OMNI API	-	2011
Kandoo	Python, C, Go and C++	U Toronto	V 1.0.0, 1.2.0	Physically distributed logically centralized	✓	✓	✓	×	Linux	JAVA RPC	-	2012
McNettle	Haskell	Yale University	V 1.0.0	Centralized	×	×	✓	×	Linux	MCNettle API	Haskell standard lib	2012
MicroFlow	C	Independent developer	V 1.0.0, V 1.5.0	Centralized	✓	×	✓	×	Linux	Socket	-	2012
Open Contrail	Python, C++ and C	Juniper	BGP, XMPP	Centralized	✓	✓	✓	✓	Linux	REST	-	2012
RouteFlow	C++	CPqD	V 1.0.0 – 1.3.0	Distributed	✓	✓	✓	×	Linux	RPC	-	2012
Ryu NOS	Python	OSRG Group, NTT laboratories	V 1.0.0, 1.1.0, 1.2.0, 1.3.0, 1.4.0 and 1.5.0	Logically Centralized	✓	✓	×	✓	Mac OS, Linux	RESTful API, ad-hoc API	LACP lib	2013
Meridian	Java	IBM	V 1.0.0, 1.3.0	Centralized	✓	✓	✓	×	Cloud Based	REST	libvirt	2013
Jaxon	Java	Independent Developer	V 1.1.0	distributed	✓	×	×	×	-	-	-	2013

programming language. ElastiCon SDN controller provides better reliability and scalability. It also provides desirable results related to consistency-based challenges. ElastiCon uses the SIGAR and RESTFUL as API, and HazelCast library for performance evaluation.

9) FLOODLIGHT [69]

Floodlight is a physically centralized architecture-based open source SDN controller developed in 2010 by Big Switch. Its performance was limited in achieving the results of reliability. However, it attains the better results of scalability and consistency. It uses Java as a programming language and operates with the Openflow version 1.1.0. Floodlight uses Java RPC, Rest, Quantum as APIs, and uses Linux, Windows, MacOS for controller evaluation. The main benefit of the Floodlight is its easy usage. The workflow of the Floodlight controller is shown in Fig.4.

TABLE 3. *(Continued.)* **A detailed summary of SDN controllers-highlighting platform, developers and detailed architecture.**

B4	Python, C	Google	V 1.0.0	Physically distributed logically centralized	✓	✓	✓	×	Linux	-	-	2013
HP VAN SDN	Java	HP	V 1.0.0	distributed	✓	✓	✓	✓	Linux	RESTful API	Open flow lib	2013
PANE	Haskell	NSF, NDSEG	V 1.0.0	Distributed	✓	✓	×	✓	Linux, MacOS	PANE API	Haskell and Nettle	2013
Trema	Ruby and C	NEC	V 1.0.0	Centralized	✓	✓	✓	×	Linux	Ad-hoc	OF lib	2013
Yanc	C++ and C	U Colorado	V 1.0.0 - 1.3.0	Distributed	✓	×	✓	×	Linux	REST	libyanc	2013
Rosemary	C	DARPA, AFRL	V 1.0.0, 1.3.0 and XMPP	Centralized	✓	✓	✓	×	Linux	Ad-hoc	Boost lib	2014
POX	Python	Nicira, Barkeley, Stanford	V 1.0.0	Centralized	✓	✓	×	×	Linux, Window and MacOS	Ad-hoc	-	2014
OVS	C	Independent Developer	OVSDB,	-	✓	×	×	✓	Linux	-	-	2014
DISCO	Java	-	V 1.0.0	Physically distributed logically distributed	✓	✓	✓	×	Linux	REST	JVM	2014
DC Fabric	JavaScript, C	OpenStack Juno	V 1.3.0	Centralized	✓	✓	✓	✓	Linux	REST	-	2014
ElastiCon	Java	NSF, IIS	V 1.0.0	Distributed	✓	✓	✓	✓	-	SIGAR, RESTful API	HazelCast lib	2014
Fleet	Python	-	V 1.0.0	Distributed	✓	✓	×	×	Linux	Ad-hoc API	Pycrypto and gmpy2	2014
MUL	C	Independent developers	V 1.0.0 - 1.4.0 and OSVDB	Centralized multi-threaded	✓	✓	✓	×	-	REST API	-	2014
Node Flow	JavaScript	Independent Developer (Dreamer Lab)	V 1.0.0	Centralized	✓	×	✓	×	JSON	NodeJS	libuv	2014
ONOS	Java	AT&T, Cisco, Ericsson, Huawei, Google	V 1.0.0, 1.3.0	Distributed	✓	✓	✓	✓	Linux, MacOS and Window	REST, neutron	-	2014
OpenIRIS	Java	ETRI	V 1.0.0, 1.3.0	Distributed	✓	×	✓	×	Linux	Loxigen, REST	-	2014
SmartLight	Java	FCT, EC FP7	V 1.3.0	Distributed	✓	✓	×	×	Linux	REST	Replication lib	2014
SDX	Python	Berkeley, Princeton	V 1.1.0	Physically distributed logically centralized	✓	×	×	✓	Linux, Windows	RESTful	-	2014
Unified Controller	-	IBM	V 1.0.0	-	✓	×	✓	✓	Linux and window	REST API	ITIL	2014
TinySDN	C	FAPESP	V 1.0.0	Centralized	✓	×	✓	×	Linux	Ad-hoc	-	2015
RunOS	C++	ARCCN	V 1.3.0	Distributed	✓	✓	✓	✓	Linux	REST API	DPDK and netmap	2015
Ravana	C++	NSF, ONR, U Princeton	V 1.1.0	Physically centralized	✓	×	×	✓	Xeon, Zookeeper	Ryu API	Parsing lib	2015
Faucet	Python	WAND Network Research Group	V 1.3.0	Centralized	✓	×	×	✓	Linux	-	-	2015

10) FAUCET [70]

Faucet is a centralized architecture-based open source SDN controller developed by WAND network Research Group in 2016. The main benefit of Faucet controller is the implementation of security policies on each access port. It uses the Python as a programming language, Linux OS as compiler for performance evaluation, and it operates with the Openflow version 1.3.0. Faucet does not achieve the desirable goals of scalability and reliability but it performs well in perspective of consistency.

11) FLEET [71]

In 2014, distributed architecture-based open source SDN controller 'Fleet' was developed, it operates with the

TABLE 3. *(Continued.)* **A detailed summary of SDN controllers - highlighting platform, developers and detailed architecture.**

ZeroSDN	C++	U Stuttgart	V 1.0.0, 1.3.0	Distributed	✓	×	×	✓	Linux	REST	ZeroMQ lib	2015
Beehive	Go	NSERC	V 1.0.0, 1.1.0 and 1.2.0	Distributed hierarchical	✓	✓	✓	✓	Linux	REST	-	2016
Floodlight	Java	Big Switch	V 1.1.0	Physically Centralized	✓	✓	×	✓	Linux, MacOS, window	Java RPC, REST, Quantum	-	2016
Loom	Erlang	Erlang Solutions	V 1.3.0, 1.4.0	Distributed	✓	×	✓	×	Linux	JSON	SDK	2016
LibFluid	C++	Open networking foundation	V 1.0.0, 1.3.0	-	✓	×	✓	×	Linux and Window	C++ API	ROFL	2016
Ravel	Python	NSA and NSF	V 1.0.0	Centralized	✓	×	✓	✓	Linux	Ad-hoc	-	2016
SNAC	C++	Nicira, Big Switch	V 1.0.0	Physically centralized	×	✓	✓	×	Linux	REST API	-	2016
OpenDaylight	Java	Cisco, Ericsson, HP, Intel, Brocade	V 1.0.0, 1.3.0	Distributed	✓	✓	✓	✓	Linux, Window and MacOS	REST, RESTCONF, XMPP	JVM	2017
POF	Java	UST China, NSFC, Huawei institute of R & D	V 1.0.0 and POF-FIS	Centralized	✓	×	✓	×	Linux	POF API	DPDK lib	2017
Ryu	Python	NTT	V 1.3.0	Logically Centralized	✓	✓	×	×	Linux	REST API	-	2017

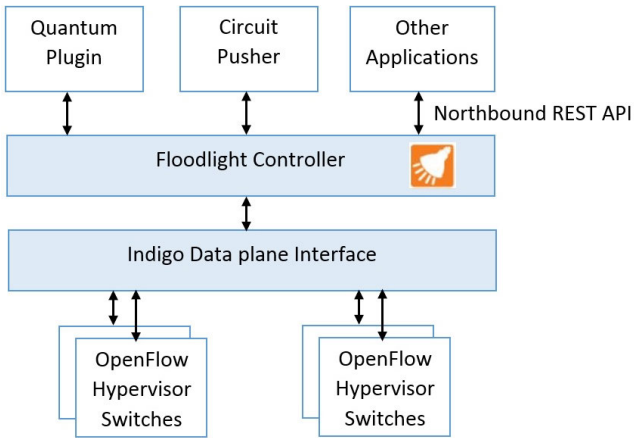

FIGURE 4. **Work flow of floodlight controller.**

Openflow version 1.0.0 and use Python as a programming language. It use Pycrypto and gmpy2 library, ad-hoc API, and Linux as OS for controller evaluation performance. Fleet vastly detect the failures as it provides better results of reliability but does not achieve the desirable results of scalability and consistency.

12) FLOWVISOR [72]

Stanford and Nicira developed a centralized architecture-based open source SDN controller 'FlowVisor' in 2009, and have use the JSON API and Linux as OS for controller performance evaluation. FlowVisor acts as transparent proxy controller between the physical switches of Openflow and operates with the Openflow versions 1.0.0 - 1.3.0. FlowVisor have use C as programming language. Its performance is limited in achieving the results of reliability and consistency. The workflow of the flowvisor controller is shown in Fig. 5.

13) HP VAN SDN [73]

In 2013, HP established a distributed architecture-based open source SDN controller, 'HP VAN SDN'. It has use JAVA as programming language and works with the Openflow version 1.0.0. HP VAN SDN controller provide better results for reliability, scalability and consistency as It uses the OpenFlow library, Linux as OS and RESTful API for performance evaluation.

14) HELIOS [74]

Distributed architecture-based an open source SDN controller 'Helios' was developed by NEC in 2012. The results of the reliability and consistency are limited by using Helios controller but provides the better results of scalability. It uses Thrift RPC library, C as programming language and Linux as OS for performance evaluation. Helios provides better results in perspective of low cost, complexity, and energy as it operates with Openflow version 1.0.0.

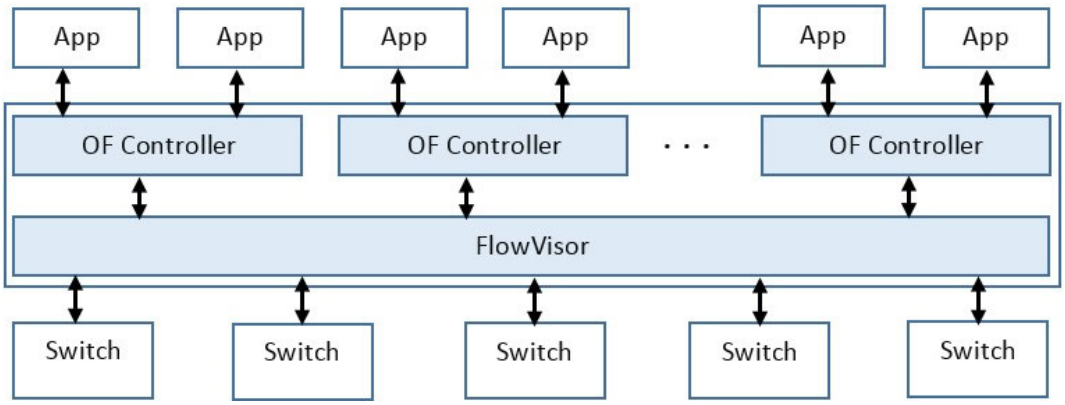

FIGURE 5. Workflow of flow visor controller.

15) HYPERFLOW [75]

In 2010, University of Toronto developed a physically distributed architecture-based open source SDN controller, 'Hyperflow'. Hyperflow operates with the Openflow version 1.0.0, have use the C++ as programming language and Boost serialization library for experiment. The advantage of HyperFlow is minimizing the cross region control traffic, also provide the better reliability and scalability however, its performance is limited in achieving the goals of consistency.

16) JAXON [76]

An open source SDN controller 'Jaxon' was developed by Independent Developer in 2013. It does not provide better results for reliability, scalability and consistency. Jaxon operates with Openflow version 1.1.0 as it uses Java as programming language.

17) KANDOO [77]

University of Toronto in 2012 developed physically distributed architecture-based open source SDN controller 'Kandoo'. Its switches can natively run local control applications. Kandoo SDN controller operates with Openflow versions 1.0.0 - 1.2.0. It uses JAVA RPC API, Python, C and C++ as programming languages and Linux as OS for experimentation. Its performance is limited with respect to consistency but provides better results in terms of reliability and scalability.

18) LOOM [78]

Distributed architecture-based open source SDN controller was developed in 2016 by Erlang Solutions. Its aim is to protect the network from the rouge applications. Loom use Erlang as programming language, SDK lib and operates with the Openflow version 1.3.0 - 1.4.0. Loom SDN controller provides good results for scalability, however, its performance is limited in reliability and consistency. It uses the JSON API and Linux as OS for performance evaluation.

19) LIBFLUID [79]

An open source SDN controller 'Libfluid' was established in 2016 by Open networking foundation. It uses the C++ as programming language. Libfluid has inflexible protocol and configurable behavior in implementation. Scalability results are achieved by using LibFluid controller, but its performance is limited in reliability and consistency. LibFluid have used Linux, window, MacOS as OS, ROFL lib and C++ API for experimentation and it works with Openflow versions 1.0.0 - 1.3.0.

20) MCNETTLE [80]

Yale University, in 2012, established an open source SDN controller 'McNettle' based on centralized architecture. It uses the Haskell as programming language, Haskell standard lib, McNettle API and Linux as OS for controller performance evaluation and it works with the Openflow version 1.0.0. The main advantage of McNettle is its high level expressive language and multicore optimization. Its performance is limited in reliability and consistency but have better results of scalability.

21) MUL [81]

MUL is an open source SDN controller based on centralized multi-threaded architecture. It was developed by Independent Developers in 2014. MUL is modular, flexible, and easy to learn. It is limited in consistency but provides better results of scalability and reliability. It uses C as programming language. MUL SDN controller works with Openflow versions 1.0.0 - V 1.4.0 and have used the REST API for experimentation.

22) MAESTRO

Maestro is a centralized architecture-based open source SDN controller established in 2010 by Rice University. Maestro have multi-threaded support and have use the JAVA as programming language. Its performance is limited in consistency but provides good results of scalability and reliability. Maestro operates with Openflow version 1.0.0. Maestro have used JVM lib, Linux, windows and Mac as OS and ad-hoc API for performance evaluation.

23) MICROFLOW [82]

MicroFlow is a centralized architecture-based open source SDN controller. It was established in 2012 by Independent

Developers, and works with the Openflow versions 1.0.0 - V 1.5.0. MicroFlow SDN controller performance is limited in reliability and consistency but it provides good results of scalability. Microflow have used Linux OS, Socket API and C as programming language for performance evaluation.

24) MERIDIAN [83]

Meridian is a centralized architecture-based open source SDN controller established by IBM in 2013. The main advantage of Meridian is the management of dynamic updates related to topology. Meridian have used libvirt, REST API and Cloud-based OS for experimentation. Meridian SDN controller provides better results of reliability and scalability. However, their performance is limited in consistency. It uses JAVA as programming language and operates with the Openflow versions 1.0.0 and 1.3.0.

25) NODEFLOW [84]

In 2014, Dreamer Lab developed a centralized architecture-based open source SDN controller called 'NodeFlow'. The NodeFlow is lightweight and efficient for real time applications. NodeFlow SDN controller performance is limited in perspective of reliability and consistency. However, it provides better results of scalability. It operates with the Openflow version 1.0.0. NodeFlow uses libuv, NodeJS API, and JavaScript as programming language and JSON for controller performance evaluation.

26) NOX [85]

Nicira and Stanford in 2008 developed an open source SDN controller 'NOX'. NOX deals with packets payload and have use the C++ and Python as programming languages. NOX SDN controller provides good results of reliability, however, it does not performs well in achieving the results of consistency and scalability. NOX uses Linux OS, and ad-hoc API for experimentation and operates with Openflow version 1.0.0.

27) NOX-MT [85]

Physically distributed architecture-based open source SDN controller 'NOX-MT' was established by Big Switch, Nicira, and University of Toronto in 2011. The main benefit of NOX-MT is the improvement of response time and throughput of NOX controller. It uses C++ as programming language, ASIO lib, ad-hoc API and Linux OS for performance evaluation. It operates with the Openflow version 1.0.0. NOX-MT SDN controller provides better result of reliability, but its performance is limited in consistency and scalability.

28) ONIX [86]

In 2010, Nicira, NTT and Google established an open source SDN controller 'ONIX'. ONIX provide solutions to almost every problem related to network management. ONIX SDN controller provide better results of reliability and scalability however, its performance is limited in consistency. It operates with Openflow version 1.1.0. It uses C++, Python, and JAVA as programming language, ONIX, NVP, NB API and third party lib for experimentation. The workflow of the ONIX Controller is shown in Fig. 6.

29) ONOS [87]

ONOS was developed by AT & T, Cisco, Ericsson, Huawei, and Google in 2014. It has a distributed architecture. The results of reliability, scalability, and consistency are achieved by using ONOS controller as it operates with the Openflow versions 1.0.0, and V 1.3.0. ONOS provide the results of throughput in perspective of performance and availability. It has used Linux, Window, Mac as OS and REST, neutron API for performance evaluation.

30) OMNI [88]

In 2011, a distributed architecture-based open source SDN controller 'OMNI' was established by Federal University of Rio de Janerio. OMNI blackuce the packet loss rate and it works with the Openflow version 1.1.0. OMNI SDN controller have use the OMNI API, Java, and Python as programming languages and Linux as OS for performance evaluation. OMNI SDN controller does not provide desirable results of reliability, scalability and consistency.

31) OPENCONTRAIL [89]

OpenContrail is a centralized architecture-based open source SDN controller developed in 2012 by Juniper. It uses C, C++ and Python as programming languages. Dynamic routing and trunking are its main advantages. OpenContrail SDN controller provide better results of reliability, scalability, and consistency. OpenContrail has used Linux as OS, REST API for experimentation and operates with the XMPP and BGP.

32) OVS [90]

OVS is an open source SDN controller developed by independent developer in 2014. Traffic filtration secures by OVS are its main advantage. OVS controller operates with OVSDB. It uses C as a programming language, and Linux as OS for experimentation. OVS SDN controller is not able to achieve the goals of scalability and reliability however, it performs well in perspective of consistency.

33) OPENDAYLIGHT [91]

In 2017, Cisco, Ericsson, HP, Intel, and Brocade established a distributed architecture-based open source SDN controller, 'OpenDaylight'. It uses JAVA as a programming language, and works with Openflow versions 1.0.0, and 1.3.0. OpenDaylight SDN controller achieves the desirable results of reliability, scalability and consistency. OpenDayLight uses JVM lib as library; REST, RESTCONF, XMPP as APIs, and Linux, Window as OS for performance evaluation.

34) OPENIRIS [92]

Distributed architecture-based open source SDN controller 'OpenIRIS' was developed by ETRI, in 2014. It uses JAVA

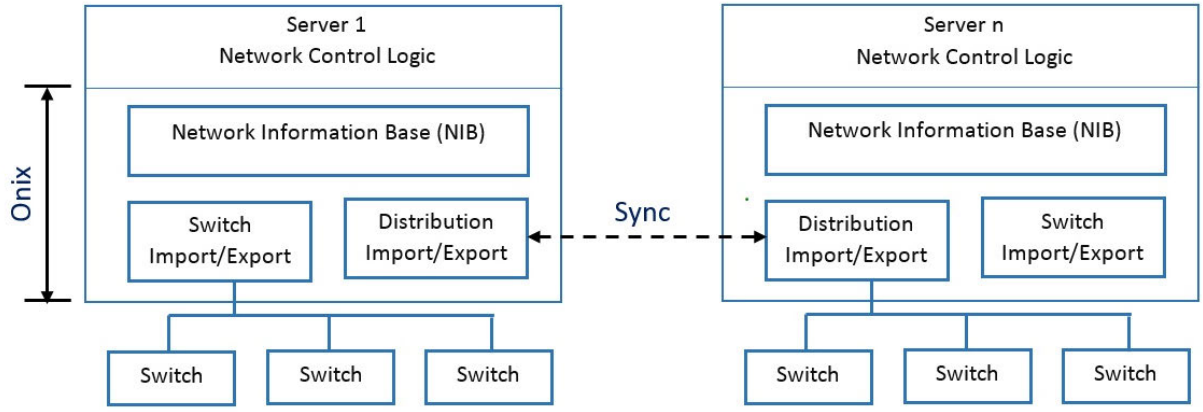

FIGURE 6. Workflow of onix controller.

as a programming language and operates with the Openflow versions 1.1.0, and V 1.3.0. The main advantage is that OpenIRIS have lower latency. OpenIRIS uses Loxigen, REST API and Linux as OS for experimentation. OpenIRIS controller provides better results of scalability however, its performance is limited in achieving the desirable results of reliability and consistency.

35) POF [93]

In 2017, University of science and technology China (USTC), NSFC, and Huawei developed a centralized architecture-based open source SDN controller, 'POF'. It operates with the Openflow version 1.0.0 and POF-FIS. It uses Java as a programming language, DPDK lib as library, POF as API, and Linux as OS for the experimentation. POF SDN controller provides better results of scalability however, its performance is limited in achieving the results of reliability and consistency.

36) PANE [94]

'PANE', an open source SDN controller, is developed by NSF and NDSEG in 2013. It uses Haskell programming language, Nettle lib as library, and PANE API and windows as OS for experimentation. PANE SDN controller provides better results of reliability and consistency, but does not provide the desirable results of scalability. It operates with Openflow version 1.0.0. The main advantage of this controller is that OMNI provides better security and performance.

37) POX [95]

An open source SDN controller 'POX' was developed by Nicira, Barkeley and Stanford, in 2014. The main advantage of using POX controller is that it easily deploy and test the SDN applications. POX controller works with Openflow version 1.0.0. It uses ad-hoc API, Python as programming language and MacOS as OS for performance evaluation. POX SDN controller achieve the results of reliability, however does not provide the desirable results of scalability and consistency.

38) PROGRAMMEABLEFLOW [2]

Physically centralized architecture-based open source SDN controller 'Programmeableflow' was developed by NEC, in 2010. Programmeableflow use the switches and other network services to control the entire infrastructure. Its performance is limited in achieving the results of consistency and scalability but, it attains the results of better reliability. The controller operates with Openflow version 1.0.0. It uses JAVA as a programming language, and control API and Linux as OS for experimentation.

39) ROSEMARY [96]

In 2014, DARPA and AFRL, established an open source based SDN controller, 'Rosemary', based on centralized architecture. It operates with Openflow versions 1.0.0, 1.3.0 and XMPP. Rosemary performance is limited in achieving the results of consistency but attains the results of better reliability and scalability. Rosemary controller uses Boost lib, ad-hoc API and Linux as OS for performance evaluation.

40) RAVEL [97]

NSA and NSF developed a centralized architecture-based open source SDN controller, 'Ravel' in 2016. Its users can modify, launch, and switch between the abstractions by using Ravel controller. Ravel operates with Openflow version 1.0.0, and have use ad-hoc API. It uses Python as a programming language, and linux as OS for experimentation. Ravel achieves the desirable results of scalability and consistency, however its performance is limited in achieving the results of reliability.

41) RYU [98]

Logically centralized architecture-based open source SDN controller, 'Ryu', was developed in 2017 by NTT. It operates with the Openflow version 1.1.0. Ryu controller performance is limited in achieving the results of scalability and consistency, but attains better results in reliability. It uses Python as programming language, Java RPC as library, Rest, Quantum

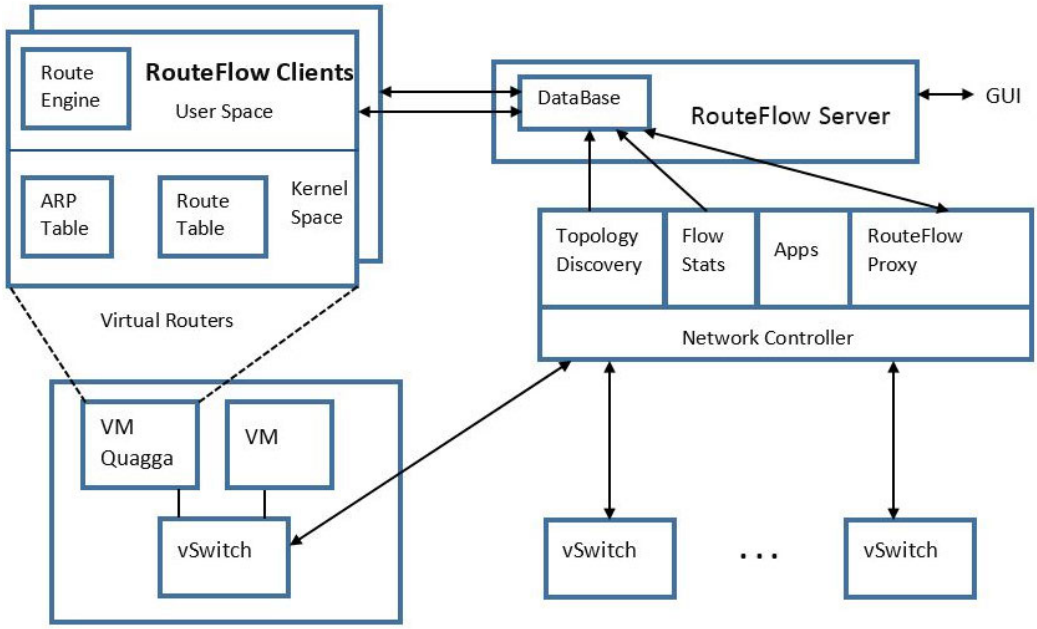

FIGURE 7. Workflow of routeflow controller.

as APIs, and Linux, Windows, MacOS as OS for controller evaluation.

42) RUNOS [99]

In 2015, ARCCN established a distributed architecture-based open source SDN controller, 'RunOS'. It uses C++ as programming language and operates with Openflow version 1.3.0. It achieves desirable results of reliability, scalability, and consistency. It uses DPDK, netmap lib as library, REST API, and Linux as OS for performance evaluation. The main benefit of using RunOS is that it reduces the equipment costs and network operation.

43) RAVANA [100]

'Ravana' is an open source SDN controller that is established by NSF, ONR, and University of Princeton in 2015. It supports multi-threaded applications. Ravana works with Openflow version 1.1.0, It uses parsing lib, Ryu API, C++as programming language, and Zookeeper for experimentation. Ravana SDN controller achieves better results of consistency, but do not achieve desirable results of scalability and reliability.

44) ROUTEFLOW [101]

RouteFlow is a distributed architecture-based open source SDN controller established by CPqD in 2012. It delivers IP routing solutions for the network. It uses C++ as programming language and works with the Openflow versions 1.1.0 - 1.3.0. It uses the RPC library and Linux as OS for performance evaluation. RouteFlow SDN controller provide better results of reliability and scalability. However, its performance is limited in achieving optimal results of consistency. The workflow of the Routeflow architecture is depicted in Fig. 7.

45) RYU NOS [47]

OSRG and NTT developed a logically centralized architecture-based an open source SDN controller 'Ryu NOS', in 2013. Ryu NOS does not require high end switches because of tunneling based isolation. It uses the LACP lib, RESTful, ad hoc API and Linux, MacOS as OS for experimentation. Ryu NOS SDN controller provide better results of reliability and consistency however, it does not achieve the desirable results of scalability. Ryu NOS operates with the Openflow versions 1.0.0 - 1.5.0.

46) SNAC [48]

In 2016, Nicira and Big Switches established a physically centralized architecture-based open source SDN controller, 'SNAC'. SNAC increases the visibility because it reports many flow level traffic details over a web UI and HTTP-based API. It operates with Openflow version 1.0.0. It uses REST API, C++as programming language, and Linux as OS for performance evaluation. SNAC SDN controller performance is limited in achieving the results of consistency but, attains better results of reliability and scalability.

47) SMARTLIGHT [49]

'SmartLight' is an open source SDN controller developed by FCT and EC FP7 in 2014. It operates with the Openflow version 1.3.0. Smartlight uses the Replication lib, REST API, Java as programming language, and Linux as OS for performance evaluation. SmartLight SDN controller achieve the results of reliability, but does not achieve the desirable results of scalability and consistency.

48) SDX [50]

In 2014, Berkeley and University of Princeton developed a physically distributed architecture-based open source SDN controller, 'SDX'. It provides remote control over the traffic. It uses Python as programming language and operates with Openflow version 1.1.0. It uses the RESTful API and Linux, Windows as OS for performance evaluation. SDX SDN controller provide better results of consistency. However, its performance is limited in achieving the desirable results of reliability and scalability.

49) TREMA [51]

NEC developed a centralized architecture-based open source SDN controller, 'Trema' in 2013. It only supports Linux-based applications. Trema operates with Openflow version V 1.0.0, and have use the Ruby and C as programming languages. It uses ad-hoc API, OF lib as library, and Linux as OS for experimentation. Trema achieves desirable results of scalability and reliability, but lacks to attain better results of consistency.

50) TINYSDN [52]

'TinySDN' is an open source SDN controller established by FAPESP in 2015. It operates with the Openflow version V 1.0.0. TinySDN uses ad-hoc API, C as programming language and Linux as OS for performance evaluation. TinySDN has the capability to perform flexible bit masking. TinySDN achieves good results of scalability, but its performance is limited in achieving the goals of scalability and consistency.

51) UNIFIED CONTROLLER [53]

IBM developed an open source SDN controller, 'Unified controller', in 2014. It operates with the Openflow version 1.0.0. Unified controller achieves the goals of scalability and consistency. However, its performance is limited to achieve sound results of reliability. It uses the REST APIs, and Linux and window as OS for performance evaluation.

52) YANC [54]

'Yanc' is a distributed architecture-based open source SDN controller established by University of Colorado in 2013. Yanc provides security, congestion control, and load balancing facility. It uses C++ and C as programming languages, and operates with the Openflow versions 1.0.0 - 1.3.0. It uses REST API, libyanc as library, and Linux as OS for performance evaluation. Yanc provides better results of scalability, but do not achieve desirable results of consistency and reliability.

53) ZEROSDN [55]

In 2015, University of Stuttgart developed a distributed architecture-based open source SDN controller, 'ZeroSDN'. ZeroSDN controller has a highly modularized and lightweight aptitude. ZeroSDN achieves desirable results of consistency, but do not attain appropriate results of scalability and reliability. ZeroSDN controller operate with the Openflow versions 1.0.0 - 1.3.0. It uses the REST API, C++ as programming language, ZeroMQ lib as library, and Linux as OS for performance evaluation.

C. CLASSIFICATION OF THREATS AND VULNERABILITIES

Security threats are spanned over three layers of SDN and two layers of their communication channel as shown in Fig. 8. Different security threats in SDN architecture are divided into 7 major groups: Data leakage, Data modification, Denial of service, Configuration issues, Compromised and the malicious applications, unauthorized access, and system-level SDN security as depicted in Fig. 9. Different examples related to these security issues are also discussed in detailed in the subsequent sections. Furthermore, a detailed description of the attacks and network security issues of SDN framework is also presented.

The security issues classified in Fig. 9 are also mapped to the SDN architecture in Fig. 10 to highlight the entity and interface impacted by the attack or vulnerability. It is worth pointing out that several of these attacks are related directly to the SDN characteristics such as unauthorized access and data modification linked with logically centralized control, malicious application linked with third party network services, DoS attack linked with open programmable interface and configuration issue threats are linked with switch management protocol characteristics.

We also mapped these potential SDN attacks to relevant security concerns.i.e. authentication, secrecy, integrity etc. as shown in Table 4.

TABLE 4. Mapping of SDN attacks to security concern.

Generic SDN Attacks	Relative Security Concern
Unauthorized Access	Authentication
Data Leakage	Confidentiality
Data Modification	Integrity
Malicious Applications	Integrity, Availability
DOS Attack	Availability
Configuration Issues	Availability

1) UNAUTHORIZED ACCESS

Unauthorized access is a sub-category of access control and subdivided into two categories: one is unauthorized or unauthenticated applications and the other one is unauthorized Controller hijacking. SDN characteristics are either described as the centralized controller or as logically centralized/distributed controller. There is a possibility in the SDN functional architecture that the data plane of network can be accessed by using several controllers. Group of controllers can be linked with the multiple source applications. One of the level of network control is that the applications can easily read or write any state of network when the applications have some information regarding the controller. If the controller or application is imitated by any attacker, the network operations can be easily manipulated and edited by the attackers.

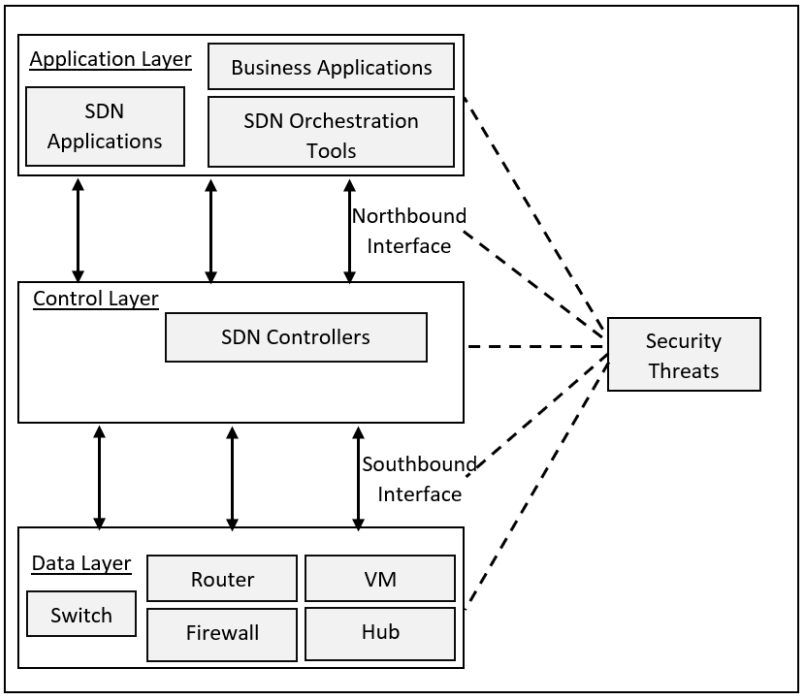

FIGURE 8. Security threats in SDN architecture: Security threats are spanned over three layers of SDN and two layers of their communication channel.

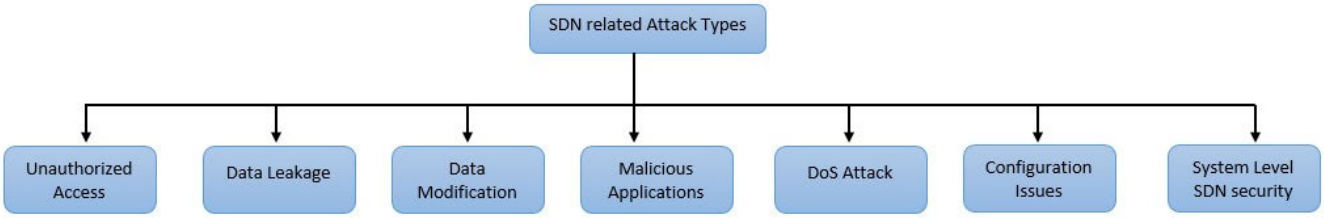

FIGURE 9. Generic SDN attacks.

2) DATA LEAKAGE

Data leakage is subdivided into three sub-categories: credential management, forwarding policy discovery, and flow rule discovery. Different type of procedures related to the packet handling are described in the Openflow specifications. These procedures are to either drop, forward , or send the packet to the controllers. By analyzing the processing time of packets, attackers can easily manage to check the type of procedure being applied to the packet. For example, the processing time of packet is less for those packets that directly pass from input to output port as compared to those packets which are forwarded to controller for further processing. Therefore, the sensitive switch configuration is easily discovered by the attacker. Extra information related to the network device can be gathered by attackers through the sets of manufactured packets. Flow request proceed to the DoS attack by the attackers when they discovered the controller directed packet type. Data leakage to the DoS attack relationship is demonstrated in [56]. In the data plane, credential storage for various networks is one of the major challenges of the SDN architecture. The attacks related to the side channel virtual machine in cloud environment have already been explained. In such an attack, important information can be extracted and identified from target by malicious virtual machines [57]. There is a possibility in SDN that the similar data can be leaked.

We can assume that different customers are allocated to each entity. So, there is a possibility that the logical network functionality can be compromised in the case when the networks and their related credential are not securely separated.

3) DATA MODIFICATION

Traffic flow in the SDN can be controlled by controller network devices. The entire system of network can be controlled whenever the controller is hijacked by the attackers. Packets can be passed from the networks to the attacker, if the attacker is able to modify the configuration

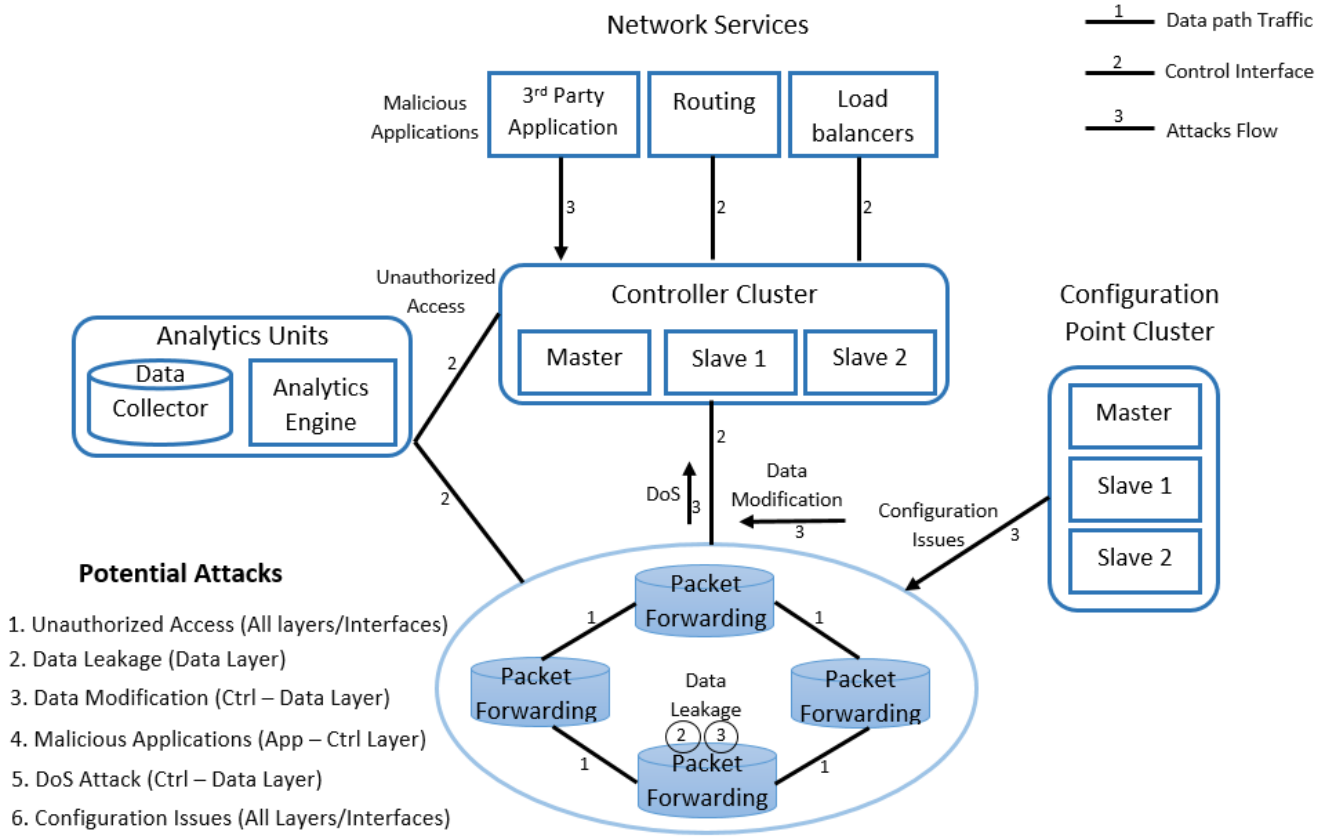

FIGURE 10. SDN potential attacks and vulnerabilities mapped to the SDN architecture to highlight the entity and interface impacted by the attack or vulnerability.

of network devices, and additional packets can be inserted by the attacker. Proper security mechanism between the components, communication channel, and the interface can be used when the intermediate entities for virtual network establishment can be introduced between the data and control plane [4]. Mutual authentication with the use of Transport Layer Security (TLS) can be described between the switches and controllers in the communication channel. Various attacks can be launched and the controllers imitate by Man-in-the-Middle (MITM) attackers when TLS is not adopted by main vendors. For example, connections can be torn down when the reset messages can be inserted and the control messages can be manipulated. Security issues should be avoided when the secure intermediate connection is established between the data and control plane. MITM attack occur when the message is interrupted between the two victims by the attackers and happens only when the communication endpoint is not authenticated. FlowVisor [58] is one of the SDN controllers in which data modification attacks can be launched on entities communication by the attackers [59].

4) COMPROMISED AND MALICIOUS APPLICATIONS

The controllers describe the key idea for data plane applications, and the architecture is also combined with SDN enabled third party applications [60]. Malicious applications on the network are as harmful as the compromised controllers. Similarly, vulnerabilities can be unintentionally injected to the system by the buggy and the poorly designed applications, such as, an application is moved into a dangerous state when an attacker manipulated the identified bug.

5) DENIAL OF SERVICE

This category is further divided into two sub categories, Switch Flow Flooding and flood communication between the controller and switch. Packets can be flooded with the controller by attackers because of the decision rule requirement and the communication between the network devices and the controller. Resources related to the limited memory available with the flow table is also a type of infrastructure-level DoS attack [61]. Rule modification and the fraudulent insertion related DoS attacks are also discussed in [62].

6) CONFIGURATION ISSUES

Three attacks have been introduced in the sub-section of Configuration issues. Firstly, those related to the policy enforcement secondly, concerning the lack of authentication techniques adoption, and lastly, related to the lack of secure provision. When the network vulnerabilities are detected, the protocols and the policies related to the network security are developed. These policies and the protocols are applied

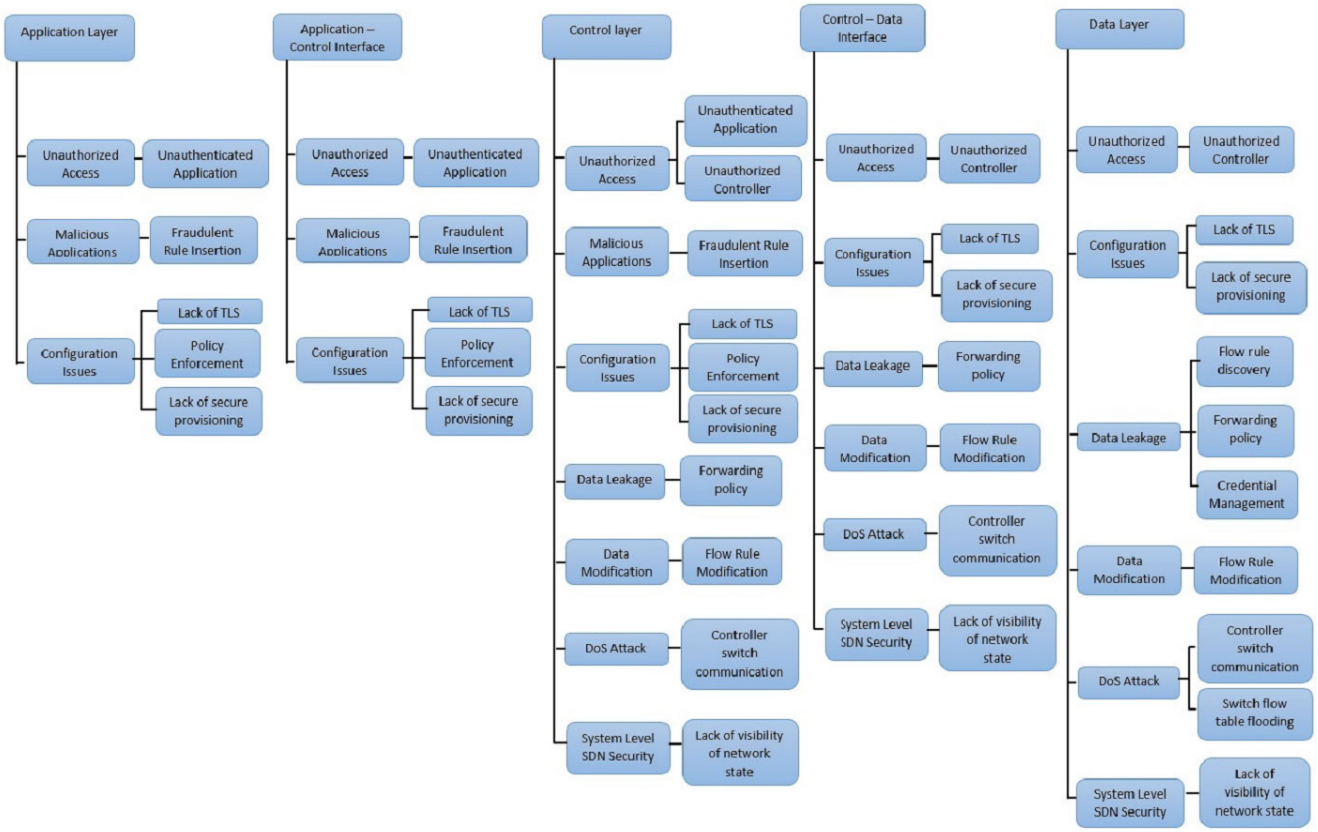

FIGURE 11. An overview of security threats and vulnerabilities related to SDN layers.

to the SDN framework interfaces and the layers. However, the policies and protocols related to SDN framework are not applied or restricted without having the knowledge the security consequences of implementation situations. Policies implementation should be applied to network operators, and is an important factor of SDN based network [32]. There is a possibility that improper and the misuse of the security features may affect all the layers of the SDN architecture. When a secure connection is disabled, different types of attacks are possible in the SDN framework that will potentially affect the network functions. Different vulnerabilities can be introduced when interfaces are opened between the network components. These vulnerabilities are not only related to the communication between the data and control plane, but also related to the compatibility between the different vendor devices. Dynamic flow policies can be created and the network can be programmed while using the SDN. Therefore, there is a possibility that the security vulnerabilities can be introduced. Policy conflicts and associated problems can be solved by ensuring consistency, and that should be formed when the policies are installed against the multiple devices and the applications. [34].

7) SYSTEM LEVEL SDN SECURITY ISSUES

Different types of SDN related security concerns are presented at the system-level. A major concern of SDN, related to the industry, is the auditing procedure. The procedure of network operation and compliance are dynamically able to provide the network devices inventory control. There are two modes in which the Openflow switches can operate: fail-standalone and the fail-secure mode [32]. From operational point of view, it is difficult for any operator to understand what type of mode is used when the connection is interrupted through the switch. In the context of auditing and the accountability, operational information retrieval and management is also one of the critical procedure. To access the resources, techniques are required to be implemented which can increase the network complexity and manageability. Integrity, confidentiality and the availability are the main properties to secure the communication between the networks. Authentication, authorization and encryption are used as supporting factors to support the properties of Confidentiality, Integrity and Availability (CIA). A network can be formed by combining the properties of CIA, in which the devices, data and communication can be protected from malicious attacks. Vulnerabilities and the challenges related to the SDN network can also be displayed in this attack category. Security attacks on different SDN layers are also shown in Fig. 11.

D. SECURITY SOLUTIONS FOR SDN CHALLENGES

As illustrated in Fig. 9, there are seven types of SDN security attacks. Different solutions were proposed to mitigate the problems of SDN security attacks. In this section, we presents

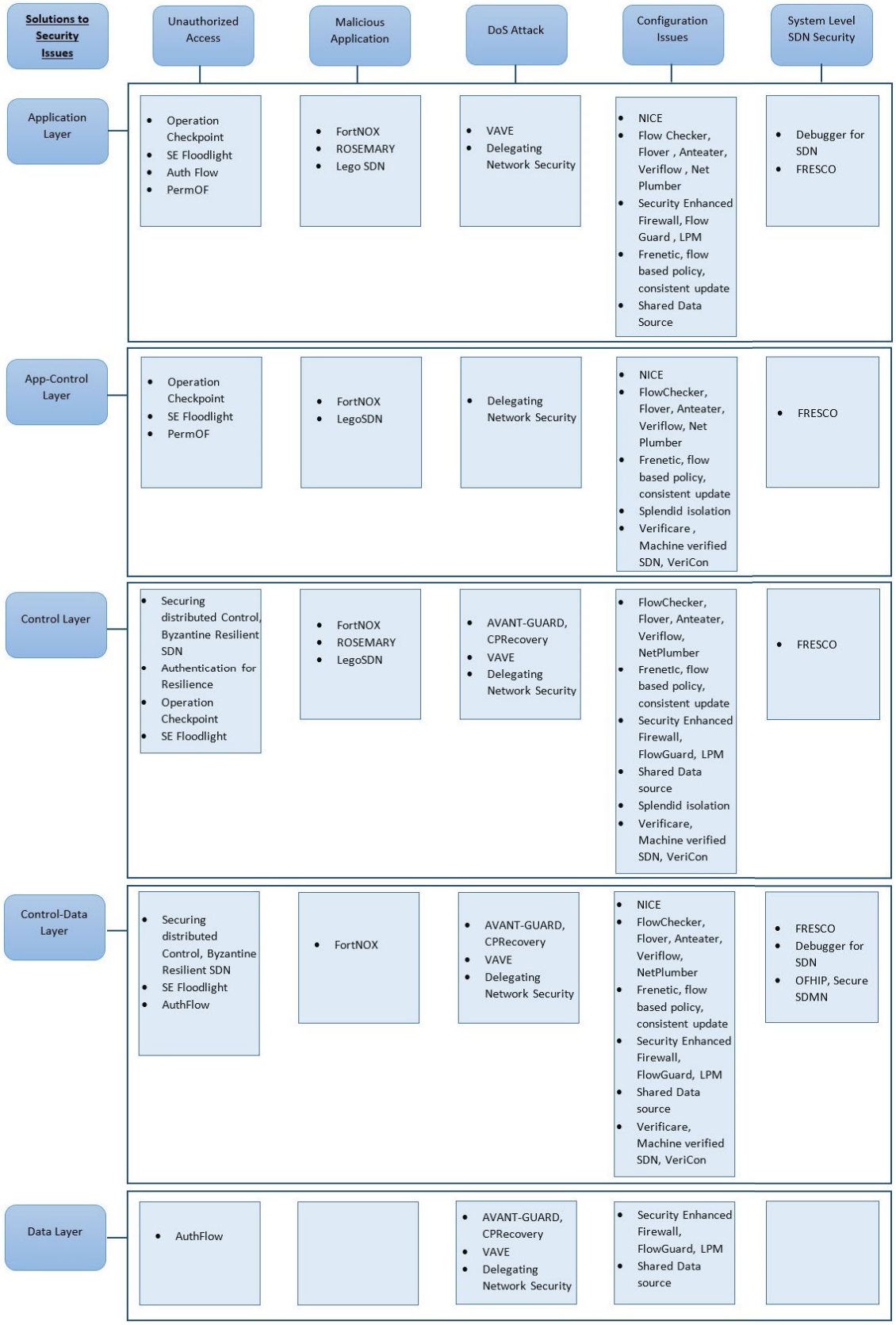

FIGURE 12. Summary of security solutions to mitigate the security threats related to different layers/interfaces (data layer, control layer, application layer, northbound interface and southbound interface) of SDN.

potential solutions to address them. Mitigating security concerns and a comparison of related works at different SDN layers are also presented in Fig. 12, that reflects the significant attention that has been recorded by the security issues in configuration, unauthorized access, and the DoS attacks. We also observed that the solution of the SDN security issues also affect the SDN layers, with the data layer being the least modified one because of the strong concentration on the software solutions. In the following sub-sections, we also provide an overview of the possible solutions relevant to SDN security.

1) UNAUTHORIZED ACCESS

A summary of the mitigation solutions to address ''Unauthorized access'' against the different SDN layers is presented in Fig.13. A Hybrid control model-based solution for the unauthorized access was proposed in [102] that improved network efficiency and reduced congestion at the controller. TLS is used to secure the central and distributed control elements. Requests between the network devices have been securely transmitted by using the signature algorithm. In their paper, system requires a centralized trust manager and for securing the system a result signature checking has been introduced. In [103], the authors presented a new mechanism, Byzantine, through which the SDN structure was secured by using the multiple controllers on each network elements. Cost efficient assignment algorithms are used to minimize the fault tolerance of Byzantine mechanism.

In [104], a new SDN setup based mechanism, with switches and the controllers, was proposed for solving the problems of authentication and authorization of applications. Working of the middle management is divided between the switch fabric and the root controller. Moreover, security is provided at the middle management controller to minimize the effect of compromised applications. In [105], the authors identified the problem of full privilege to every application without providing any protection. The PermOF-based solution was proposed by the authors to implement the set of permissions at the API level. By using their mechanism, the network can be protected from the control plane attacks. The permission system concept was also extended in [106], where the authors presented mechanism of Operation Checkpoint, which was implemented and designed on the SDN controller floodlight. Initially, a set of permissions was defined by the authors through which controller and the applications can be subscribed with each other. Profile of the malicious activities for the application layer based attacks were maintained by using the unauthorized operation logs. A new mechanism SE floodlight was proposed by Stanford Institute in [107], [108]. Northbound API was digitally authenticated in SE floodlight mechanism and security enforcement kernel was introduced. SEK digitally verifies the administrator at run time by presigning the java class applications. A host-level access control, AuthFlow, solution was proposed for the unauthorized access in [109], where an authenticator, Openflow controller and the radius server were used to implement the mechanism Authflow. AuthFlow is used for the authentication and the access control of the applications. Traffic was controlled by the controller based on the response of authentication.

2) MALICIOUS/COMPROMISED APPLICATIONS

Connection establishment and authentication must be performed between the application and the controller before exchanging the messages to avoid the implementation of any type of malicious application. Different types of solutions were proposed to mitigate the malicious applications' concern. A summary of related solutions are presented in Fig. 14 that are proposed to address the ''Malicious and Compromised Applications'' issues at different SDN layers. FortNOX, a security enforcement kernel-based solution, was proposed to solve the issue of malicious applications in [110]. Role-based authentication was implemented in FortNOX mechanism for authorization of Openflow applications. Flow rule insertion with the possible conflicts can be handled by enforcement engine of FortNOX and the authorization of the author security is the main factor that is used to accept or reject the rule. FortNOX is also able to detects the flow rules that are in conflict with each other. If the new flow rule is requested with a lower priority or a higher priority, then the existing flow rule can be ignored or can be replaced respectively. Priority enforcement and the identification of application-based issues were not resolved in FortNOX.

A secure and robust network operating system, ROSEMARY [96], was proposed to address malicious applications. It can be used to strengthen the security of control plane. Micro-NOS based architecture was proposed by author as a solution to solve the issue of malicious applications. Control plane can be protected from any malicious application and the vulnerability by running the Openflow application within the instance of ROSEMARY. The solution separates the NOS and the network applications. Network resources used by any application can be monitored and controlled. LegoSDN an isolation layer between the SDN applications based solution was proposed by authors in [111], to avoid crashing of SDN application and the SDN controllers. Automatic updates can be enforced by using the network wide transaction system of LegoSDN whereas, crash events can be detected and overcome by fault tolerance layer. Third party network services' issues were also protected by LegoSDN.

3) DOS ATTACK

Decoupling of control and data plane can lead to DoS attacks on the controller. Different solutions have been proposed to solve the issue of DoS attack on SDN layers. A short summary of the solutions are presented in Fig. 15 to mitigate the SDN security issue of ''DoS Attack'' against the different SDN layers.

The DoS attack workflow is shown in Fig. 16. From the starting point, a packet will be received by the openflow

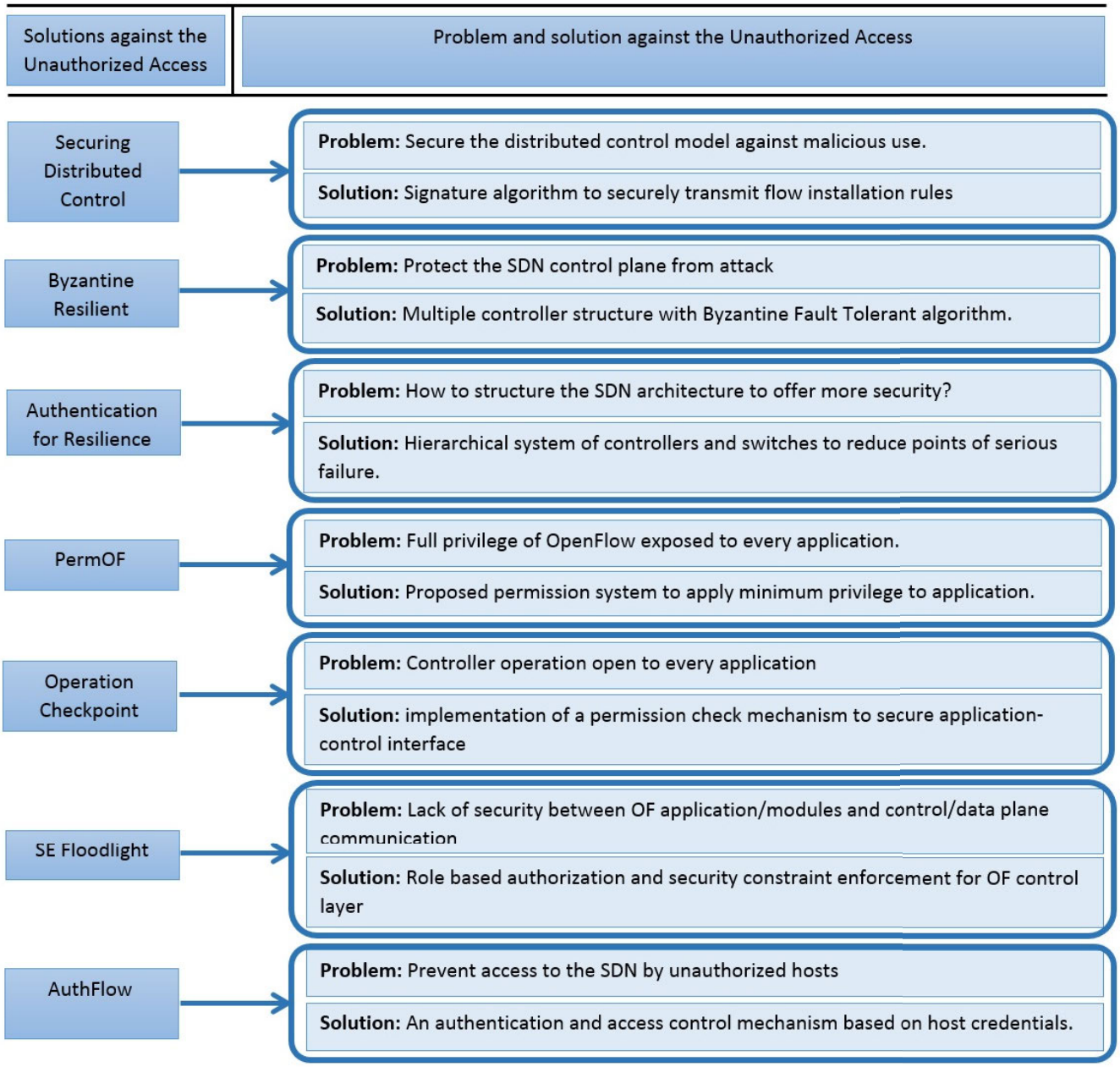

FIGURE 13. Summary of security solutions to mitigate the SDN security issue of unauthorized access: securing distributed control [102], byzantine resilient [103], authentication for resilience [104], PermOF [105], operation checkpoint [106], SE FloodLight [107], [108], AuthFlow [109].

switch and will be checked against the flow rule of the table. If there is no match in the table for that particular packet, switch will send a request to SDN controller to create a new rule for that flow. Afterwards, SDN controller will respond by sending a new forwarding rule for that flow. On the other hand, if there is a match in the table for packet, the switch will forward that packet according to the existing flow rules. Therefore, packet passing the NAT gateway will move toward the online services. In an alternative scenario, the switch will mirror each received packet to the IDS. The IDS will then analyze the packets to assess the occurrence of DoS attack. If the packet comes under the category of DoS attack, the IDS notifies the SDN controller about the attack. Furthermore, SDN controller will send the blocking flow rule to the switch. The switch will then mitigated the attack by removing all the packets as per the new installed blocking rule.

AVANT GUARDA, a control and data plane communication based solution was proposed in [112]. It uses a migration tool to limit the requests that were sent to the control plane. The Migration tool was used to remove the TCP sessions that failed at the data plane before sending any notification to the control plane in TCP SYN flood attack. Network flexibility

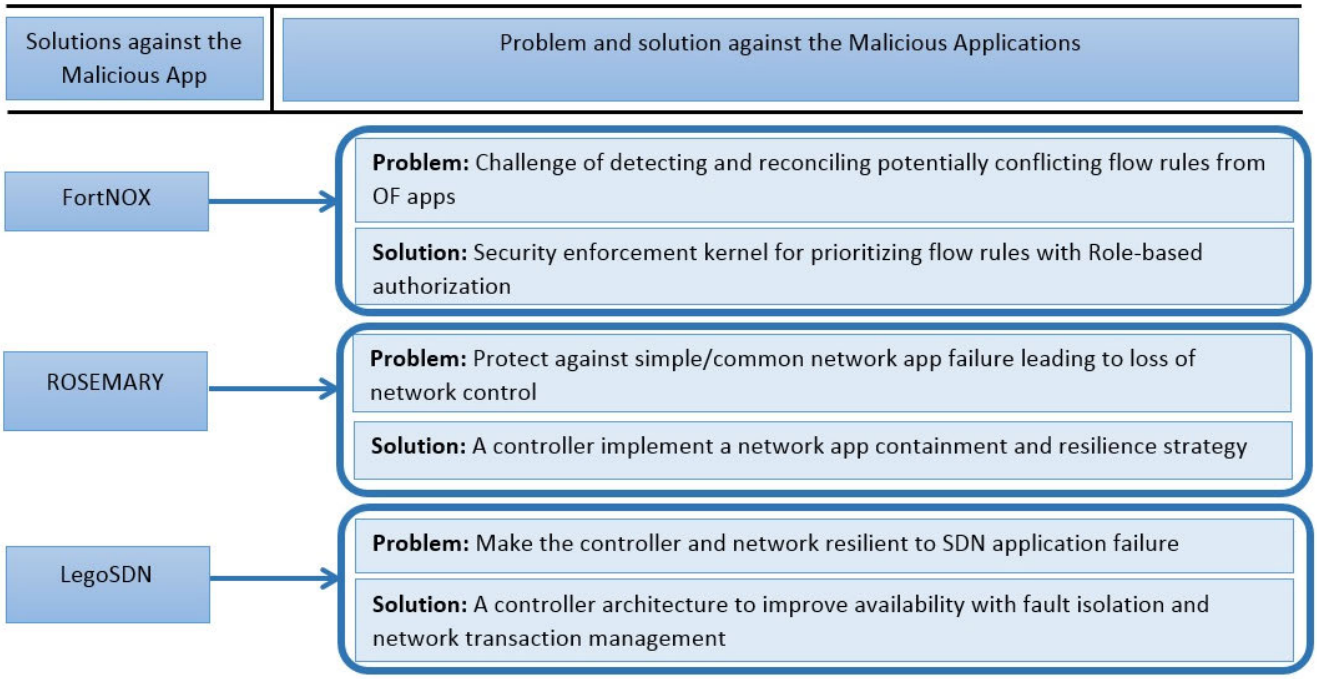

FIGURE 14. Summary of security solutions to mitigate the SDN security issue of malicious and compromised application; FortNOX [110], ROSEMARY [96], LegoSDN [111].

can be improved by providing the solution of CPRecovery in [113]. When an external attacker overcomes the NOS, CPRecovery provides a continuous transition from failed controllers to the backup. A defensive approach, based on source address validation, to protect the system against the IP Spoofing was proposed in [114], called VAVE. Protection against the IP Spoofing can be provided by doing the traffic analysis of SDN and dynamically updating the rule. If an incoming packet is not matched against the rule of Openflow, the source address can be validated by sending that Openflow switch to the controller. And, if any type of spoofing is detected, the traffic can be stopped from the source address by installing the rules at the switch. VAVE can be used for protection of the data layer against the DoS attack. In [57], ident++ protocol based solution was proposed for the DoS attack challenges produced by congestion in the network management.

Users and the end host play the role of network administrators in the implementation of network security applications. SDN attacks can be overcome in the solution by using the SDN characteristics. The possibility of a DoS attack can be decreased by using the distributed control and the dynamic flow table. In [115], an extended version of the IEEE 802.1X-based solution, FlowNAC, was proposed to solve the issue of SDN-related DoS attacks. Incoming frames from the user can be categorized and evaluated using the FlowNAC solution. In [116], a cloud-based solution, CloudWatcher, was proposed to monitor the security of the cloud SDN environment. DaMask is presented as a solution for SDN-related DoS attacks [117]. DaMask is divided into two subsections, DaMask-D and DaMask-M, and is used for detection and mitigation purposes, respectively. In DaMask-D, the dataset shift problem can be addressed by using the graphical model of probabilistic interference. Countermeasures for different attacks can be matched in DaMask-M, and the new countermeasure can also be regular black in it. The workflow of the solution DaMask is shown in Fig. 17. By using a combination of a correlator and monitor, a new solution was proposed by Chin et al., in [118], for protection against a SYN flood attacks. The working of correlator and the monitor are similar as the former is used in DaMask-M and the latter in DaMask-D. Another solution, based on the traffic flow algorithm, FlowTrApp, was proposed in [119]. In FlowTrApp, the flow duration and the flow rate were used to control the traffic flow. The attacker was not blocked on the first attempt but was instead blocked for sending malicious traffic that was not matched with any traffic pattern. An IDS-based solution was proposed by Hu et al. [120], in which network attacks can be detected by using the event processing engine. The engine is comprised of an event bus, channel, sub controller, and hyper controller. Sub-controllers can be organized by the hyper-controller, and any type of malicious traffic can be detected and sent through the event bus and event channel. Skowyra proposed a new solution, learning IDS, in [121]. It provides a response to malicious intents by changing the network states. Server survival time can be extended in a DoS attack by introducing the balancing method proposed by

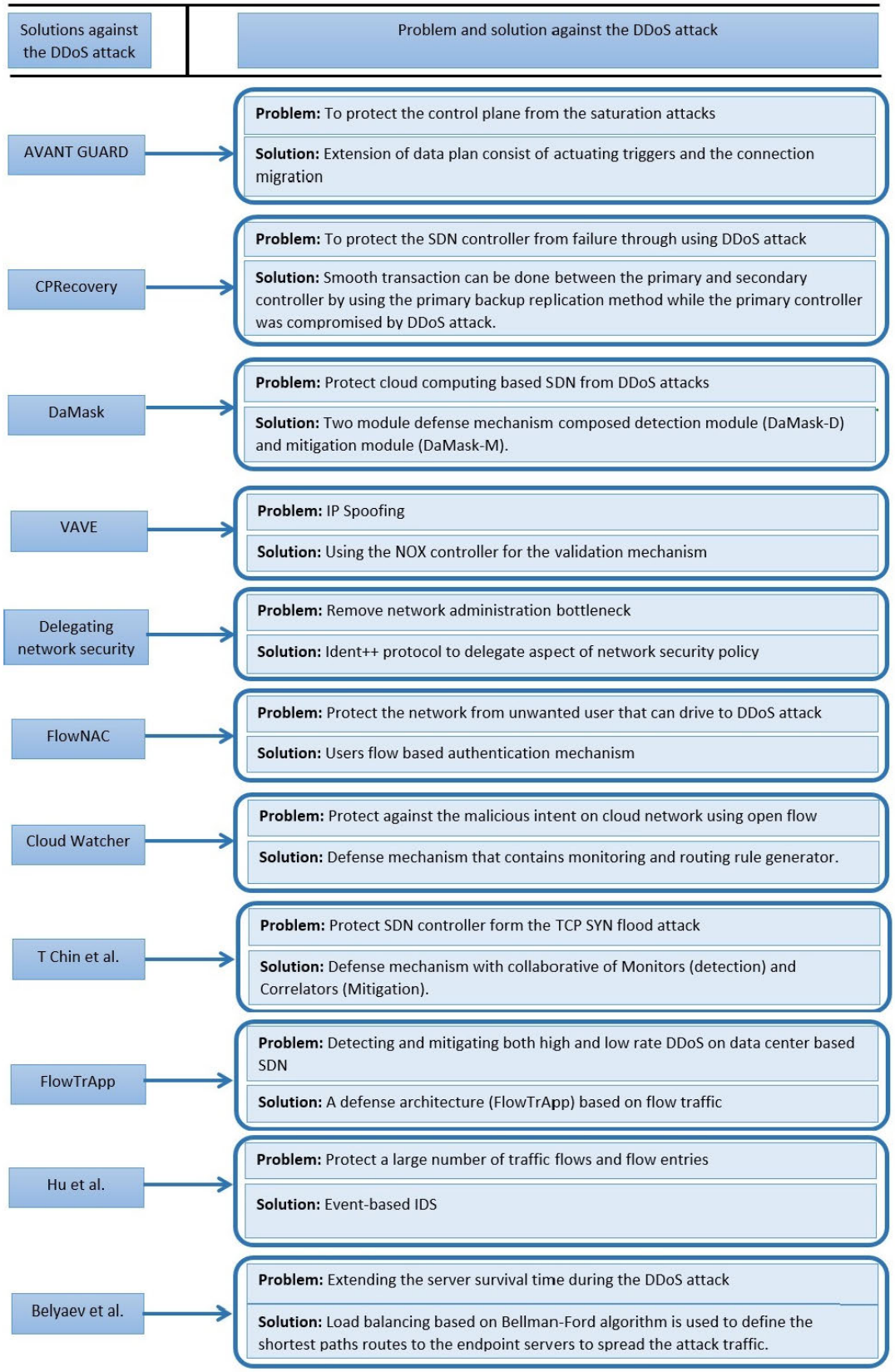

FIGURE 15. Summary of solutions to mitigate SDN security issues of DoS; AVANT-GUARD [112], CPRecovery [113], VAVE [114],Delegating Network Security [57], FlowNAC [115], CloudWatcher [116], DaMask [117], Chin et al. [118], FlowTrApp [119], Hu et al. [120] Learning IDS [121], Belyaev et al. [122], SHDA [123].

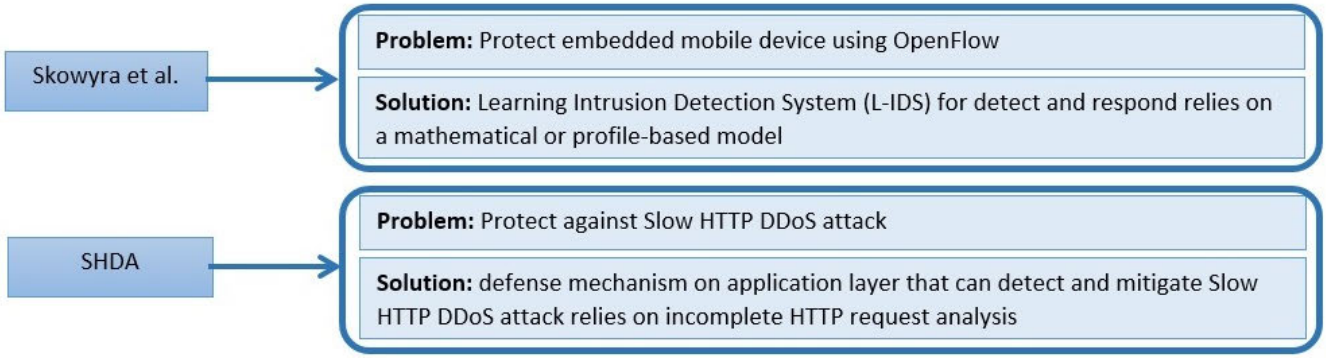

FIGURE 15. *(Continued.)* **Summary of solutions to mitigate SDN security issues of DoS; AVANT-GUARD [112], CPRecovery [113], VAVE [114],Delegating Network Security [57], FlowNAC [115], CloudWatcher [116], DaMask [117], Chin et al. [118], FlowTrApp [119], Hu et al. [120] Learning IDS [121], Belyaev et al. [122], SHDA [123].**

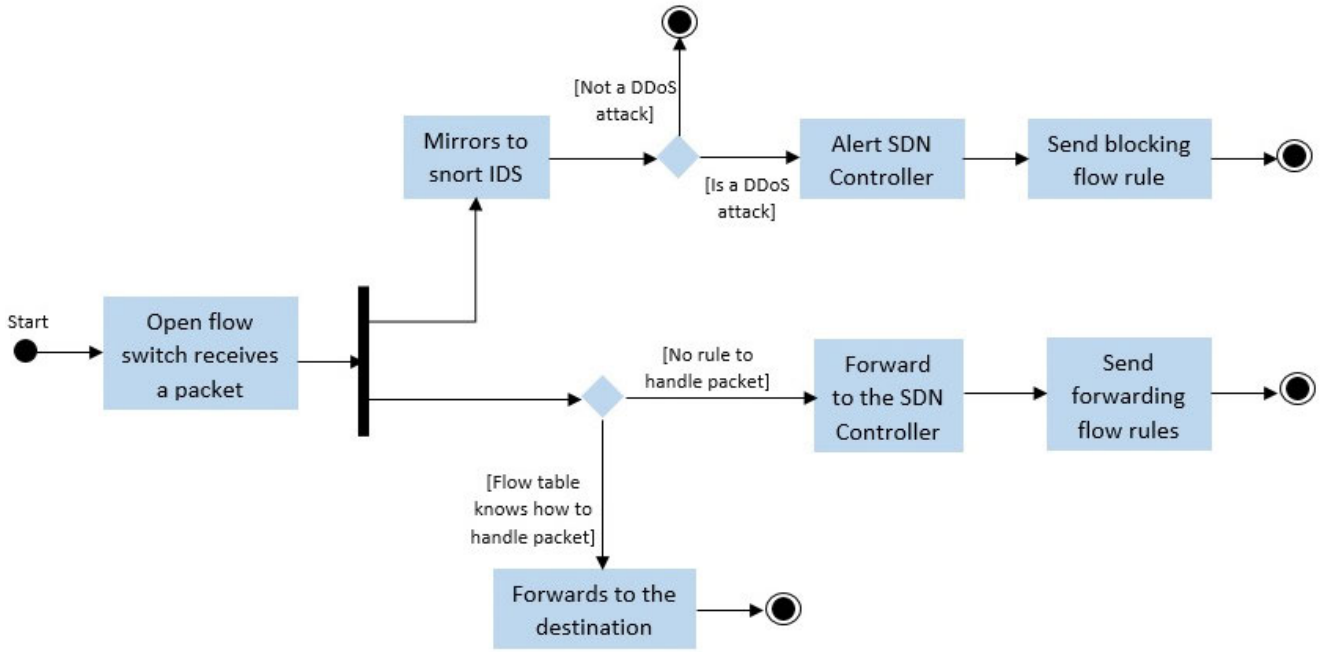

FIGURE 16. **Activity diagram showing the sequence of DoS attack on controllers.**

Belyaev et al., in [122]. The routing table can be overcome by a balancing algorithm whenever there is an attack on the server. Traffic can be distributed by using the shortest path route of the Bellman-Ford algorithm. A solution for the HTTP-based DoS attack on the application layer was proposed by SHDA in [123].

4) SYSTEM LEVEL SDN SECURITY

Different types of solutions have been proposed to securely implement the SDN paradigm. A summary of the solutions to mitigate the issue of "System level SDN Security" against the different SDN layers is presented in Fig. 18. A Network debugging based prototype solution was proposed in [124]. The solution provides root causes and bug identification using event chain reconstruction. Network debugging and auditing requirements can be supported by using this solution. Host identity protocol and IPSec encapsulating security protocol were combined to propose a new solution for OFHIP in [125]. As a benefit of OFHIP, secure mobility can be enabled, and OpenFlow problems can be avoided. Mobility can be supported between the networks by enabling the Openflow switches in OFHIP and securely changing their IP addresses. Mobile network-based SDN issues were solved by authors in [126], in which the issues, threats, and attacks were considered in solutions along with the OFHIP work. IP-based attacks can be presented, analyzed, and detected by using the architecture of secure control channels. A secure solution was proposed in which the control channel was combined with the mobile network to protect against spoofing, eavesdropping, and replay attacks. The security enforcement kernel FortNOX, along with the application development framework-based solution FRESCO, was proposed in [127]. Design and the development of the security modules are the core ideas behind the solution of FRESCO, which can be

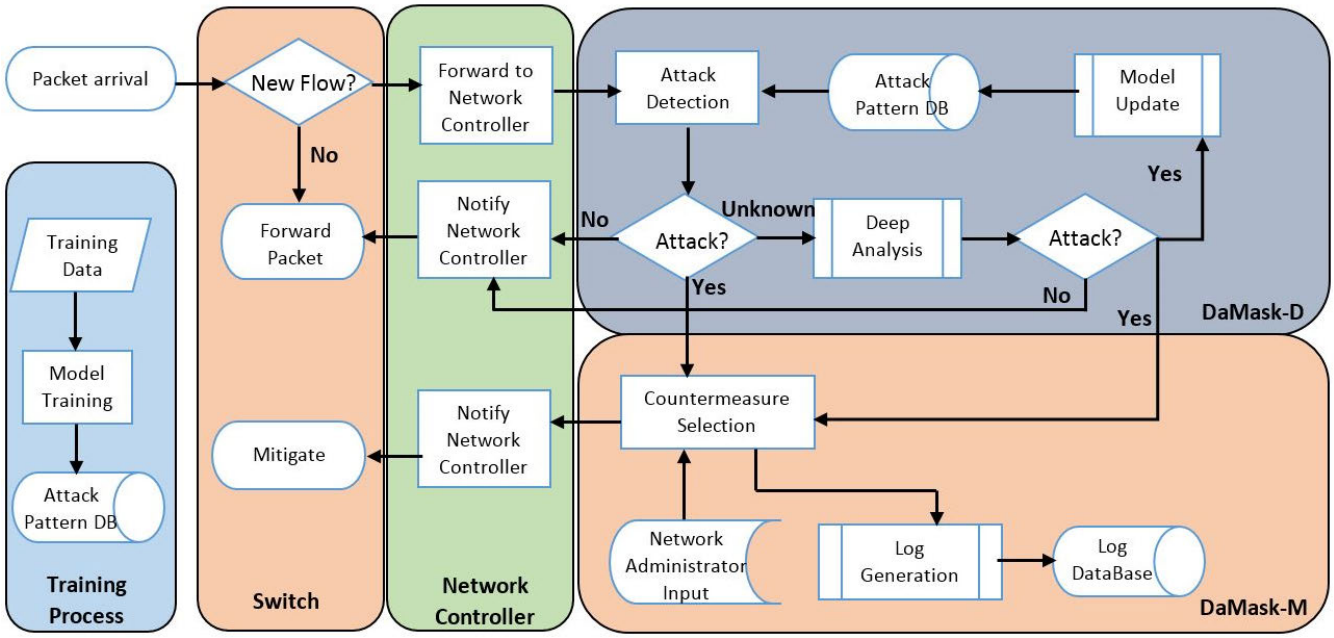

FIGURE 17. Detailed work flow of DaMask.

combined with the OpenFlow applications. Threats can be detected and mitigated by deploying the reusable module library. Solutions were presented to identify the different environments in which the SDN was installed.

5) CONFIGURATION ISSUES

Three types of attacks have been introduced in the sub-section of this threat category related to policy enforcement, lack of authentication technique adoption, and lack of secure provision. Different solutions have been proposed to solve the issue of SDN configuration. The proposed solution was further divided into five sections as follows, also depicted in Fig. 19:

1) Detection of network errors
2) Real time Policy Checking
3) Language based resolution
4) Consistent Abstraction/Network View
5) Formal Verification Method

a: DETECTION OF NETWORK ERRORS

Application correctness can be checked by combining the symbolic execution with modeling checks in a solution called NICE [128]. This solution is used when a network state reaches a conflicting position. A new mechanism, Flow Checker, for the network errors was proposed in [129], in which the reconfiguration of an intra-switch of flow table can be tested by using the binary decision diagram. Flover, proposed in [130], verifies flow policies using modulo theories and assertion sets. The response time of the controller can be improved by using batch mode with the SDN controller Flover. Configuration problems in the network can be detected by using a static analysis approach in Anteater [131]. Verification of the data plane can be done by using these solutions against the configuration issues. The execution time of the corresponding system and the time required to find the problems can range from seconds to hours.

b: REAL TIME POLICY CHECKING

A new solution VeriFlow was proposed in [132], in which the verification of flow rules can be made before they reach any network state. Loops can be detected from the path and the routing tables by modeling a graph in VeriFlow. The main objective of using this solution is to detect real-time network traffic, and the performance can be checked within seconds. Real-time networking is a collection of hardware and software devices that respond in real time. The NetPlumber tool [133] can be used to check the policies of real-time applications. The authors previous work based on the header space analysis was improved in NetPlumber to validate real-time updates using dependency graphs. An SDN-based firewall was proposed in [134]. In this proposal, the resolution algorithm and conflict detection can be developed by applying the HSA methods. SDN firewalls can be potentially bypassed by using the switches in the flow entries. The flow space can be checked, and the flow rules can be authorized by using the flow graph in the solution. The flow path of the conflicting part can be blocked by using the denying flow rule. A prototype of the solution was implemented in the firewall application of the floodlight controller. The major limitation of this solution is that it only works on rewriting the actions of the flow. Violations

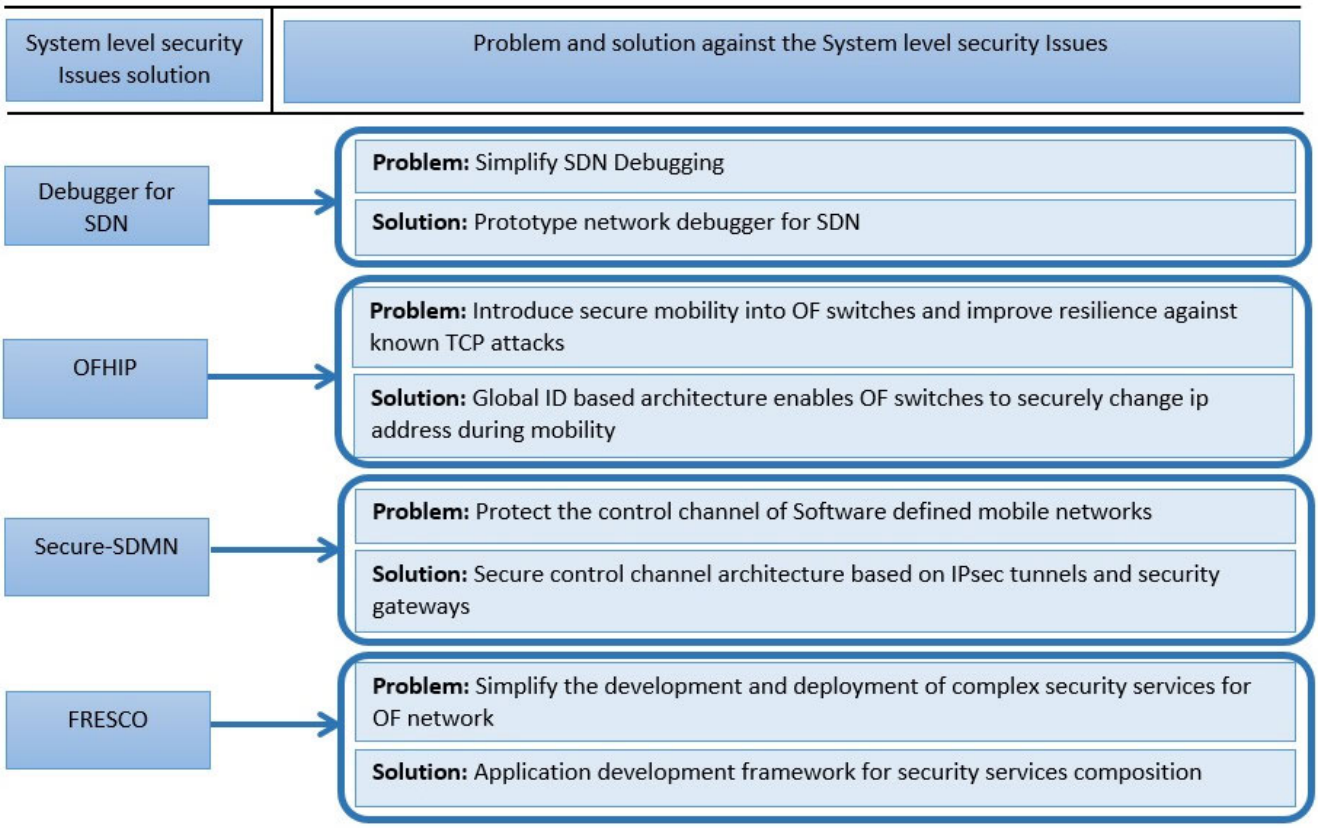

FIGURE 18. Summary of solutions to mitigate the system level SDN security; debugger for SDN [124], OFHIP [125], Secure SDMN [126], FRESCO [127].

of the firewall policies can be solved by using the solution of FlowGuard, proposed in [135] and [136], through which the network flow paths can be checked when the updates are performed. Violations related to the real time and the automatic resolution was also supported in the solution of FlowGuard. Five different strategies were used to achieve the goal of policy checking: rejection of the flow, removal of the flow, rejection can be updated, dependency breaking, and blockage of packets. Gaps between the network and the theoretical model can be linked by conducting further deployment experiments.

c: LANGUAGE-BASED RESOLUTION

Conflicts of the policies resolved by designing northbound API-based solution as proposed in Frenetic [137]. Network switches were programmed using the centralized controller methodology. Initially, the flow rules were installed in the switches by using the instructions of the controller, and later, the flow rules were changed to the policies at run time. Flow rule-based policy was implemented in the NOX controller in [138]. Access control in NOX was enforced on the external sources in the network isolation's. Policy conflict resolution was introduced between the control and application layers by providing verification and validation at the controller level. Policy conflicts were also avoided by simplifying the network programming.

d: CONSISTENT ABSTRACTION/NETWORK VIEW

Uncertainty of the configuration issues related to SDN was solved by discussing the abstraction for the flows and the per-packet consistency in [139]. The purpose of using the per-packet was that every packet would either use the new policies or the old policies, but not both at the same time. An overview of the per-packet consistency was described by using the per-flow consistency. A layer-based solution for policy management was proposed in [140]. The dependencies of the intra-table were solved at the data layer, inter-applications' at the control layer, and inter or intra-flow rules' at the application layer. A huge set of dependencies on the multiple layers was solved by dealing with the corresponding flow rules. Scalability for large networks or applications can be maintained by using conflict resolution policies. A mis-configuration of the network state can be detected by performing dedicated processing at the switch level. In [141], a new solution was proposed for the network view in which the performance of the SDN configuration was not compromised. An architecture based on the shared data was proposed by authors rather than using and implementing the techniques of policy conflict resolution.

e: FORMAL VERIFICATION METHOD

Different solutions have been proposed under the category of formal verification methods. The basic purpose is to

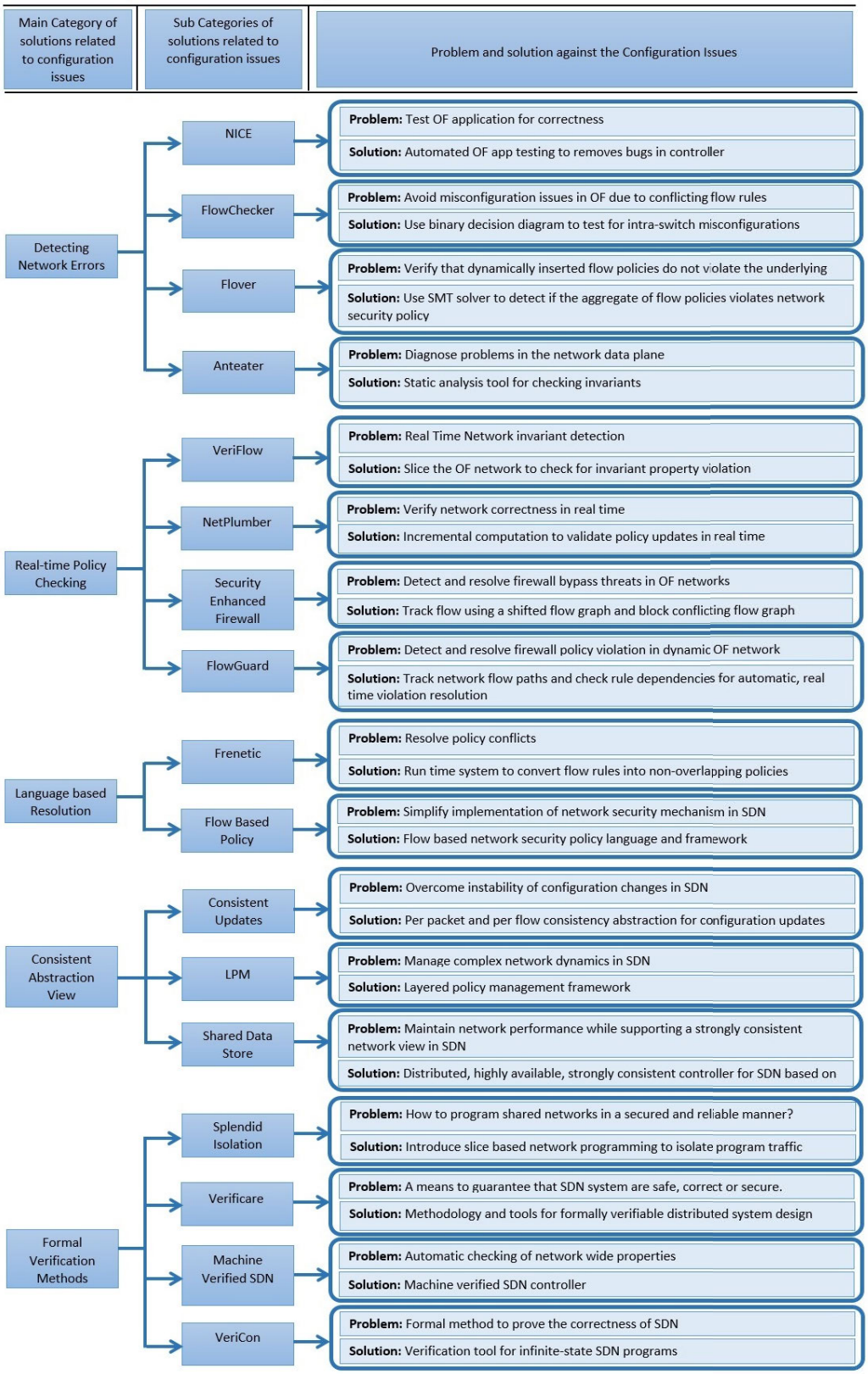

FIGURE 19. Summary of solutions to mitigate configuration issues related to security threats; NICE [128], FlowChecker [129], Flover [130], Anteater [131], Veriflow [132], NetPlumber [133], Security Enhanced Firewall [134], FlowGuard [135], [136], Frenetic [137], Flow based policy [138], Consistent updates [139], LPM [140], Shablack Data Store [141], Splendid Isolation [142], Verificare [143], Machine Verified SDN [144], VeriCon [23].

secure and save the control plane from the operating environment. A solution,'splendid isolation' was proposed by authors in [142]. Its basic purpose is to verify the traffic that flows around the network slices. Network slices can be defined by using the programming abstraction. The work used FlowViSor controller as a transparent proxy between the switches and the controller, but the solution was unable to provide security or proof of correctness for the Flowvisor. A verification tool-based solution, Verificare, was proposed by authors in [143]. Verificare is a combination of verification and the development of network correctness, convergence, and network-related mobility characteristics. An operational model of an openflow-based solution was proposed by the authors in [144]. The run-time system and the verified compiler were developed by using the solution, and the first machine-verified controller for the SDN was also implemented by using the method in [144]. Infinite state safety for the SDN program was verified by using the VeriCon solution in [23]. The main topic of the research is verification issues and network state consistency. Detection and verification tools were developed in the real-time network. The future research is based on designing these detection and verification tools for SDN deployment instead of using them for the OpenFlow-based implementation.

V. DISCUSSION AND OPEN RESEARCH CHALLENGES

A. DISCUSSION

- As shown in Table 1, we provided a consolidated overview of SDN architecture, classification of controllers, comparison of controllers, and open flow visualizations, along with their security attacks and proposed defense mechanisms.
- Presented a qualitative comparison among different SDN controllers based on different features, i.e., programming language, developer, open-flow version, architecture. reliability, scalability, consistency, compiler, API, and library. So, a researcher or reader has the information about controllers in one place instead of searching it on a browser, which is time-consuming. There are dozens of controllers that can be used, but there is no best controller; it all depends on the user requirements. The recommendation for the beginner is to use the POX and floodlight that will give them an idea of Openflow-SDN. ONOS and ODL have the latest features. Ryu can be used for prototyping because it is very straightforward and easy to program. ONOS can be used if anyone wants to get closer to telco deployments, which are a bit more complex but more realistic. OpenDaylight is too complex as a first step, as its architecture is too generalized, and it is difficult to model new modules if anyone is new to SDN.
- We have highlighted security vulnerabilities in Fig. 11 and their security solutions in Fig. 12 for the application, control, and data plane of the SDN. The most vulnerable component in SDN architecture is the control plane. As a result, the vulnerability of controllers has already been explored and studied from various perspectives, including controller protection from applications, controller scalability and availability, resilience, and security from DoS and DDoS attacks. Although security programs are designed and introduced, the security of the applicationÂ itself is a challenge. Furthermore, the optional use of TLS and DTLS in Openflow violates connectivity confidentiality between controllers and switches. With the gradual implementation of SDN technologies, it is very likely that new security threats will arise.

B. OPEN RESEARCH CHALLENGES

There are still potential and vital research areas that need attention before SDN can be commercially deployed. Some of the related research challenges are:

1) SDN STANDARDIZATION

Openflow is the actual source on which the SDN is implemented. Another SDN framework was released by IETF [145]. Network Functions Virtualization (NFV) was proposed to the ETSI Industry Specification Group (ISG) for NFV, which is highly complementary to SDN. A comprehensive comparison must be conducted among the different SDN implementations. Different types of projects use the same method to solve the same problem in the control layer, but a reliable solution has not yet been implemented. Therefore, different working groups, like the ONF and CSA alliance, should work together to design a standard protocol that will work for both the southbound and northbound interfaces.

2) ENHANCEMENT OF COMMUNICATION

TTLS is optional while its version is still not specified. It is not available for the northbound interface and is demanded to be explored. The lack of TLS usage may allow an attacker to attack the controller and launch various types of attacks. TLS can also be used for mutual authentication to validate each component's identity and prevent attackers from linking switches and rogue controllers.

3) SDN IMPLEMENTATION

Network security is one of the critical challenges related to SDN that requires serious attention. The integrity of data flow is not safe between SDN controllers and switches, such as an attacker can corrupt the network by acting as an SDN controller. Therefore, new strategies need to be introduced to provide security in the SDN environment. Moreover, we need to implement a standard programming language that will work for all instead of using different programming languages. One more drawback related to the SDN architecture is the storage and overhead, as a longer header is used for matching in the Openflow version 1.0-1.6. The performance of the SDN is also affected when the proxy is used excessively between the controller and the switching devices.

4) MACHINE LEARNING - AN INSTRUMENT TO AUGMENT SDN SECURITY

Recently, machine learning (ML) and data mining techniques have played an essential role in the detection and classification of intrusion attacks. Several machine learning studies have been conducted in different domains [30], [36]. However, only a few of these are implemented on SDN. There are many issues that can influence machine learning models' performance, such as the feature selection methods, the data set, etc. However, there is no related work that performs a systematic analysis of these machine-learning techniques. It is important to perform a methodical analysis of the popular approaches for detecting attacks in SDN and provide bench-marking analysis of traditional machine-learning-based approaches. Also, it is needed to provide a feature analysis of the input feature space of the dataset and recommendations for a reliable intrusion detection system.

5) SDN INTEGRITY

There are two bridge layers: the northbound and southbound interfaces. The basic responsibilities of the northbound inter-faces are relatively unclear, and it is also the least explored part of the SDN architecture. Learning from the southbound interface development, it is important to resolve the issues related to the communication between the controller and network applications. There are different APIs that are used in controller implementation. Therefore, desirable flexibility can be achieved by implementing them within the software system.

VI. CONCLUSION

Software-defined networking, the new revolutionary network-working technology, has emerged as a new paradigm for managing heterogeneous networks ranging from small homes to enterprises' networks. The logically centralized and distributed control plane and programmability offer a great opportunity to improve network security by implementing new mechanisms to detect and mitigate the various threats and also enable security as a service in the SDN paradigm. However, there are boundless security problems with each layer of SDN. To acquaint readers with the knowledgeable insight of controllers and to present a detailed picture of the security challenges of SDN and security solutions presented till date, this survey endeavors to fill in the research gaps as presented in Table 1 and provide all in one package. We also highlighted the SDN architecture, controllers, security threats, and different types of attacks present at different layers of the SDN. In addition, we have also presented some research solutions to address some of the security issues introduced by the SDN, i.e., how potential damage from a malicious application can be controlled. Work on these issues is being encouraged by the increasing security focus of industry-sponsored standardization and research groups. Finally, the paper concludes with a few open research challenges related to the SDN implementation, deployment, and standardization.

ACKNOWLEDGMENT

The authors would like to thank the National University of Sciences and Technology for providing the adequate facilities to carry out this research.

REFERENCES

[1] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 333–354, 1st Quart., 2018.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[3] P. Patel, D. Bansal, L. Yuan, A. Murthy, A. Greenberg, D. A. Maltz, R. Kern, H. Kumar, M. Zikos, H. Wu, C. Kim, and N. Karri, "Ananta: Cloud scale load balancing," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, Aug. 2013, pp. 207–218.

[4] S. Natarajan, A. Ramaiah, and M. Mathen, "A software defined cloud-gateway automation system using OpenFlow," in *Proc. IEEE 2nd Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2013, pp. 219–226.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, and J. Zolla, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[6] J. M. Smith, D. J. Farber, C. A. Gunter, S. M. Nettles, D. C. Feldmeier, and W. D. Sincoskie, "SwitchWare: Accelerating network evolution," CIS Dept., Univ. Pennsylvania, Philadelphia, PA, USA, White Paper, 1996.

[7] O. Filip. (2017). *The BIRD Internet Routing Daemon*. [Online]. Available: http://bird.network.cz/

[8] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM SIGOPS Operating Syst. Rev.*, vol. 33, no. 5, pp. 217–231, Dec. 1999.

[9] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. V. D. Merwe, "Design and implementation of a routing control platform," in *Proc. 2nd Int. Conf. Netw. Syst. Design Implement.*, vol. 2, 2005, pp. 15–28.

[10] V. Varadharajan, K. Karmakar, U. Tupakula, and M. Hitchens, "A policy-based security architecture for software-defined networks," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 897–912, Apr. 2019.

[11] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 2, pp. 493–501, Mar. 2019.

[12] D. S. Rana, S. A. Dhondiyal, and S. K. Chamoli, "Software defined networking (SDN) challenges, issues and solution," *Int. J. Comput. Sci. Eng.*, vol. 7, no. 1, pp. 884–889, Jan. 2019.

[13] M. Alsaeedi, M. M. Mohamad, and A. A. Al-Roubaiey, "Toward adaptive and scalable OpenFlow-SDN flow control: A survey," *IEEE Access*, vol. 7, pp. 107346–107379, 2019.

[14] A. A. Abbasi, A. Abbasi, S. Shamshirband, A. T. Chronopoulos, V. Persico, and A. Pescapè, "Software-defined cloud computing: A systematic review on latest trends and developments," *IEEE Access*, vol. 7, pp. 93294–93314, 2019.

[15] R.-H. Hwang, V.-L. Nguyen, and P.-C. Lin, "StateFit: A security framework for SDN programmable data plane model," in *Proc. 15th Int. Symp. Pervasive Syst., Algorithms Netw. (I-SPAN)*, Oct. 2018, pp. 168–173.

[16] Y. Yu, X. Li, X. Leng, L. Song, K. Bu, Y. Chen, J. Yang, L. Zhang, K. Cheng, and X. Xiao, "Fault management in software-defined networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 349–392, 1st Quart., 2019.

[17] L. Zhu, M. M. Karim, K. Sharif, F. Li, X. Du, and M. Guizani, "SDN controllers: Benchmarking & performance evaluation," 2019, *arXiv:1902.04491*.

[18] P. Raghav and A. Dua, "Enhancing flow security in Ryu controller through set operations," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, Dec. 2017, pp. 1265–1269.

[19] I. H. Abdulqadder, D. Zou, I. T. Aziz, and B. Yuan, "Modeling software defined security using multi-level security mechanism for SDN environment," in *Proc. IEEE 17th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2017, pp. 1342–1346.

[20] K. Raghunath and P. Krishnan, "Towards a secure SDN architecture," in *Proc. 9th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2018, pp. 1–7.
[21] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3259–3306, 4th Quart., 2018.
[22] M. Z. Abdullah, N. A. Al-awad, and F. W. Hussein, "Performance comparison and evaluation of different software defined networks controllers," *Int. J. Comput. Netw. Technol.*, vol. 6, no. 2, pp. 36–41, May 2018.
[23] T. Ball, N. Bjørner, A. Gember, S. Itzhaky, A. Karbyshev, M. Sagiv, M. Schapira, and A. Valadarsky, "VeriCon: Towards verifying controller programs in software-defined networks," in *Proc. 35th ACM SIGPLAN Conf. Program. Lang., Design Implement.*, 2014, pp. 282–293.
[24] Z. Zhang, H. Li, S. Dong, and L. Hu, "Software defined networking (SDN) research review," in *Proc. Int. Conf. Mech., Electron., Control Autom. Eng.*, 2018, pp. 291–300.
[25] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, 1st Quart., 2015.
[26] L. Wang and D. Wu, "SecControl: Bridging the gap between security tools and SDN controllers," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, 2017, pp. 11–31.
[27] S. H. Yeganeh and Y. Ganjali, "Beehive: Simple distributed programming in software-defined networks," in *Proc. Symp. SDN Res.*, Mar. 2016, pp. 1–4.
[28] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Comput. Netw.*, vol. 72, pp. 74–98, Oct. 2014.
[29] I. Alsmadi and D. Xu, "Security of software defined networks: A survey," *Comput. Secur.*, vol. 53, no. 1, pp. 79–108, Sep. 2015.
[30] S. Dev, B. Wen, Y. H. Lee, and S. Winkler, "Ground-based image analysis: A tutorial on machine-learning techniques and applications," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 79–93, Jun. 2016.
[31] C. Bouras, A. Kollia, and A. Papazois, "Teaching 5G networks using the ONOS SDN controller," in *Proc. 9th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2017, pp. 312–317.
[32] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 623–654, 1st Quart., 2016.
[33] W. Li, W. Meng, and L. F. Kwok, "A survey on OpenFlow-based software defined networks: Security challenges and countermeasures," *J. Netw. Comput. Appl.*, vol. 68, pp. 126–139, Jun. 2016.
[34] K. Slavov, D. Migault, and M. Pourzandi, "Identifying and addressing the vulnerabilities and security issues of SDN," *Ericsson Technol. Rev.*, vol. 92, no. 7, pp. 1–12, 2015.
[35] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.
[36] C. S. Nwosu, S. Dev, P. Bhardwaj, B. Veeravalli, and D. John, "Predicting stroke from electronic health records," in *Proc. 41st Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2019, pp. 5704–5707.
[37] A. Feghali, R. Kilany, and M. Chamoun, "SDN security problems and solutions analysis," in *Proc. Int. Conf. Protocol Eng. (ICPE) Int. Conf. New Technol. Distrib. Syst. (NTDS)*, Jul. 2015, pp. 1–5.
[38] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.
[39] D. Chourishi, A. Miri, M. Milic, and S. Ismaeel, "Role-based multiple controllers for load balancing and security in SDN," in *Proc. IEEE Canada Int. Humanitarian Technol. Conf. (IHTC)*, May 2015, pp. 1–4.
[40] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," 2014, *arXiv:1406.0440*.
[41] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, Oct. 2010, p. 19.
[42] *Principles and Practices for Securing Software Defined Networks*, Open Netw. Found., Palo Alto, CA, USA, 2015.
[43] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Comput. Netw.*, vol. 71, pp. 1–30, Oct. 2014.
[44] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.
[45] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 493–512, 1st Quart., 2014.
[46] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *Proc. 9th Central Eastern Eur. Softw. Eng. Conf. Russia*, Oct. 2013, p. 1.
[47] *Ryu NOS: SDN Controllers*. Accessed: Nov. 19, 2013. [Online]. Available: https://osrg.github.io/ryu/
[48] *OpenFlow Controller: SNAC (Simple Network Access Control)*. Accessed: Sep. 2011. [Online]. Available: https://groups.geni.net/geni/raw-attachment/wiki/GEC9DemoSummary/SNAC-poster-gec9-final.pdf
[49] F. Botelho, A. Bessani, F. M. V. Ramos, and P. Ferreira, "SMaRtLight: A practical fault-tolerant SDN controller," 2014, *arXiv:1407.6062*.
[50] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "SDX: A software defined Internet exchange," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 551–562, 2014.
[51] Thomas. (2012). *Trema: SDN Controller*. [Online]. Available: https://trema.github.io/trema/
[52] B. T. de Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *Proc. IEEE Latin-America Conf. Commun. (LATINCOM)*, Nov. 2014, pp. 1229–1237.
[53] S. Racherla, D. Cain, S. Irwin, P. Ljungstrøm, P. Patil, and A. M. Tarenzio, *Implementing IBM Software Defined Network for Virtual Environments*. Brazil: IBM BlackBooks, 2014.
[54] M. Monaco, O. Michel, and E. Keller, "Applying operating system principles to SDN controller design," in *Proc. 12th ACM Workshop Hot Topics Netw.*, Nov. 2013, pp. 1–7.
[55] *ZeroSDN: SDN Controller*. [Online]. Available: https://zerosdn.github.io/
[56] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Aug. 2013, pp. 165–166.
[57] J. Naous, R. Stutsman, D. Mazieres, N. McKeown, and N. Zeldovich, "Delegating network security with more information," in *Proc. 1st ACM Workshop Res. Enterprise Netw.*, Aug. 2009, pp. 19–26.
[58] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium*, vol. 1, p. 132, Oct. 2009.
[59] V. T. Costa and L. H. M. K. Costa, "Vulnerability study of FlowVisor-based virtualized network environments," in *Proc. 2nd Workshop Network Virtualization Intelligence Future Internet*, Rio de Janeiro, Brazil, 2013.
[60] Hewlett Packard Company. (2014). *SDN Dev Center: Unlock Network Innovation*. [Online]. Available: https://www.hp.com/go/ sdndevcenter
[61] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.
[62] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 151–152.
[63] D. Erickson, "The Beacon OpenFlow controller," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Aug. 2013, pp. 13–18.
[64] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *Proc. IEEE Netw. Operations Manage. Symp. (NOMS)*, May 2014, pp. 1–4.
[65] *DC-Fabric Controller*. Accessed: 2019. [Online]. Available: https://wiki.openstack.org/wiki/DCFabric-neutron-plugin
[66] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 351–362, Aug. 2010.
[67] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, 2011.

[68] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. R. Kompella, "ElastiCon; An elastic distributed SDN controller," in *Proc. ACM/IEEE Symp. Architectures Netw. Commun. Syst. (ANCS)*, Oct. 2014, pp. 17–27.

[69] V. B. Harkal and A. A. Deshmukh, "Software defined networking with floodlight controller," *Int. J. Comput. Appl.*, vol. 975, p. 8887, Jul. 2016.

[70] J. Bailey and S. Stuart, "Faucet: Deploying SDN in the enterprise," *Queue*, vol. 14, no. 5, pp. 54–68, Oct. 2016.

[71] S. Matsumoto, S. Hitz, and A. Perrig, "Fleet: Defending SDNs from malicious administrators," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, Aug. 2014, pp. 103–108.

[72] *FlowVisor: SDN Controller*. Accessed: 2009. [Online]. Available: https://searchnetworking.techtarget.com/definit

[73] *Virtual Application Networks SDN Controller*, Hewlett Packard, Palo Alto, CA, USA, 2016.

[74] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 339–350, 2011.

[75] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw.*, 2010, pp. 1–3.

[76] (2013). *Jaxon*. [Online]. Available: http://jaxon.onuos.org/

[77] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Aug. 2012, pp. 19–24.

[78] *LOOM: SDN Controller*. Accessed: 2016. [Online]. Available: https://wiki.sdn.ieee.org/display/sdn/LOOM

[79] A. Vidal. (2015). *LibFluid: SDN Controller*. [Online]. Available: https://wiki.sdn.ieee.org/display/sdn/libfluid

[80] A. Voellmy and J. Wang, "Scalable software defined network controllers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 289–290, Sep. 2012.

[81] *OpenMul: SDN Controller*. Accessed: 2015. [Online]. Available: http://www.openmul.org/

[82] R. Narayanan, S. Kotha, G. Lin, A. Khan, S. Rizvi, W. Javed, H. Khan, and S. A. Khayam, "Macroflows and microflows: Enabling rapid network innovation through a split SDN data plane," in *Proc. Eur. Workshop Softw. Defined Netw.*, Oct. 2012, pp. 79–84.

[83] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, "Meridian: An SDN platform for cloud network services," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 120–127, Feb. 2013.

[84] *Node Flow: SDN Controller*. Accessed: 2019. [Online]. Available: https://www.oreilly.com/library/view/software-defined-networking-with/9781783984282/a6393111-31f5-4de6-bc44-b926ac108e44.xhtml

[85] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, Jul. 2008.

[86] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *Proc. OSDI*, vol. 10, 2010, pp. 1–6.

[87] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. OConnor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an open, distributed SDN OS," in *Proc. ACM 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 1–6.

[88] D. M. F. Mattos, N. C. Fernandes, V. T. da Costa, L. P. Cardoso, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte, "OMNI: OpenFlow Management infrastructure," in *Proc. Int. Conf. Netw. Future*, Nov. 2011, pp. 52–56.

[89] *Open Contrail: SDN Controller*. Accessed: 2019. [Online]. Available: https://www.sdxcentral.com/networking/sdn/definitions/juniper-contrail-controller/

[90] *OVS: SDN Controller*. Accessed: 2014. [Online]. Available: https://www.openvswitch.org

[91] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2014, pp. 1–6.

[92] B. Lee, S. H. Park, J. Shin, and S. Yang, "IRIS: The OpenFlow-based recursive SDN controller," in *Proc. 16th Int. Conf. Adv. Commun. Technol.*, Feb. 2014, pp. 1227–1231.

[93] S. Li, D. Hu, W. Fang, S. Ma, C. Chen, H. Huang, and Z. Zhu, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, no. 2, pp. 58–66, Mar. 2017.

[94] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory networking: An API for application control of SDNs," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 327–338, Oct. 2013.

[95] S. Kaur, J. Singh, and N. S. Ghumman, "Network programmability using POX controller," in *Proc. ICCCS Int. Conf. Commun., Comput. Syst.*, vol. 138, 2014, p. 70.

[96] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A robust, secure, and high-performance network operating system," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 78–89.

[97] A. Wang, X. Mei, J. Croft, M. Caesar, and B. Godfrey, "Ravel: A database-defined network," in *Proc. Symp. SDN Res.*, Mar. 2016, p. 5.

[98] *Ryu: SDN Controller*. Accessed: 2015. [Online]. Available: https://osrg.github.io/ryu/

[99] A. Shalimov, S. Nizovtsev, D. Morkovnik, and R. Smeliansky, "The Runos OpenFlow controller," in *Proc. 4th Eur. Workshop Softw. Defined Netw.*, Sep. 2015, pp. 103–104.

[100] N. Katta, H. Zhang, M. Freedman, and J. Rexford, "Ravana: Controller fault-tolerance in software-defined networking," in *Proc. 1st ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, Jun. 2015, p. 4.

[101] A. Vidal, F. Verdi, E. L. Fernandes, C. E. Rothenberg, and M. R. Salvador, "Building upon RouteFlow: A SDN development experience," in *Proc. 31st Simpsio Brasileiro de Blackes Computadores (SBRC)*, 2013, pp. 879–892.

[102] O. M. Othman and K. Okamura, "Securing distributed control of software defined networks," *Int. J. Comput. Sci. Netw. Secur.*, vol. 13, no. 9, pp. 5–14, 2013.

[103] H. Li, P. Li, S. Guo, and S. Yu, "Byzantine-resilient secure software-defined networks with multiple controllers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 695–700.

[104] D. Yu, A. W. Moore, C. Hall, and R. Anderson, "Authentication for resilience: The case of SDN," in *Proc. Int. Workshop Secur. Protocols*, 2013, pp. 39–44.

[105] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for OpenFlow applications," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Aug. 2013, pp. 171–172.

[106] S. Scott-Hayward, C. Kane, and S. Sezer, "OperationCheckpoint: SDN application control," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols*, Oct. 2014, pp. 618–623.

[107] OpenFlowSec. *Security Enhanced Floodlight*. Accessed: 2014. [Online]. Available: https://www.openflowsec.org

[108] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, "Securing the software defined network control layer," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 1–15.

[109] D. M. Ferrazani Mattos and O. C. M. B. Duarte, "AuthFlow: Authentication and access control mechanism for software defined networking," *Ann. Telecommun.*, vol. 71, nos. 11–12, pp. 607–615, Dec. 2016.

[110] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Aug. 2012, pp. 121–126.

[111] B. Chandrasekaran and T. Benson, "Tolerating SDN application failures with LegoSDN," in *Proc. 13th ACM Workshop Hot Topics Netw.*, Oct. 2014, p. 22.

[112] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 413–424.

[113] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in *Proc. IEEE Netw. Operations Manage. Symp.*, Apr. 2012, pp. 933–939.

[114] G. Yao, J. Bi, and P. Xiao, "Source address validation solution with OpenFlow/NOX architecture," in *Proc. 19th IEEE Int. Conf. Netw. Protocols*, Oct. 2011, pp. 7–12.

[115] J. Matias, J. Garay, A. Mendiola, N. Toledo, and E. Jacob, "FlowNAC: Flow-based network access control," in *Proc. 3rd Eur. Workshop Softw. Defined Netw.*, Sep. 2014, pp. 79–84.

[116] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic clouds? Networks (or: How to provide security monitoring as a service in clouds?)," in *Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2012, pp. 1–6.

[117] B. Wang, Y. Zheng, and W. Lou, "DDoS attack protection in the era of cloud computing," in *Proc. Int. Conf. Comput. Netw.*, 2015, pp. 659–664.

[118] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "An SDN-supported collaborative approach for DDoS flooding detection and containment," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2015, pp. 659–664.
[119] C. Buragohain and N. Medhi, "FlowTrApp: An SDN based architecture for DDoS attack detection and mitigation in data centers," in *Proc. 3rd Int. Conf. Signal Process. Integr. Netw. (SPIN)*, Feb. 2016, pp. 519–524.
[120] Y.-L. Hu, W.-B. Su, L.-Y. Wu, Y. Huang, and S.-Y. Kuo, "Design of event-based intrusion detection system on OpenFlow network," in *Proc. 43rd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2013, pp. 1–2.
[121] R. Skowyra, S. Bahargam, and A. Bestavros, "Software-defined IDS for securing embedded mobile devices," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2013, pp. 1–7.
[122] M. Belyaev and S. Gaivoronski, "Towards load balancing in SDN-networks," in *Proc. Int. Sci. Technol. Conf., Modern Netw. Technol. (MoNeTeC)*, 2014, pp. 1–6.
[123] K. Hong, Y. Kim, H. Choi, and J. Park, "SDN-assisted slow HTTP DDoS attack defense method," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 688–691, Apr. 2018.
[124] N. Handigol, B. Heller, V. Jeyakumar, D. Maziéres, and N. McKeown, "Where is the debugger for my software-defined network?" in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Aug. 2012, pp. 55–60.
[125] S. Namal, I. Ahmad, A. Gurtov, and M. Ylianttila, "Enabling secure mobility with OpenFlow," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, Nov. 2013, pp. 1–5.
[126] M. Liyanage, M. Ylianttila, and A. Gurtov, "Securing the control channel of software-defined mobile networks," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2014, pp. 1–6.
[127] S. W. Shin, P. Porras, V. Yegneswara, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular composable security services for software-defined networks," in *Proc. 20th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2013, pp. 1–16.
[128] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, "A NICE way to test OpenFlow applications," in *Proc. Symp. Netw. Syst. Design Implement.*, 2012, p. 10.
[129] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures," in *Proc. 3rd ACM Workshop Assurable Usable Secur. Configuration*, Oct. 2010, pp. 37–44.
[130] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Model checking invariant security properties in OpenFlow," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2013, pp. 1974–1979.
[131] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King, "Debugging the data plane with anteater," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 290–301, Oct. 2011.
[132] A. Khurshid, W. Zhou, M. Caesar, and P. Godfrey, "VeriFlow: Verifying network wide invariants in real time," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 467–472, 2012.
[133] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real time network policy checking using header space analysis," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement.*, 2013, pp. 99–112.
[134] J. Wang, Y. Wang, H. Hu, Q. Sun, H. Shi, and L. Zeng, "Towards a security-enhanced firewall application for OpenFlow networks," in *Proc. Int. Symp. Cyberspace Saf. Secur.*, 2013, pp. 92–103.
[135] H. Hu, G. J. Ahn, W. Han, and Z. Zhao, "Towards a reliable SDN firewall," in *Proc. Open Netw. Summit*, Santa Clara, CA, USA, 2014.
[136] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: Building robust firewalls for software-defined networks," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, Aug. 2014, pp. 97–102.
[137] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," *ACM SIGPLAN Notices*, vol. 46, no. 9, pp. 279–291, 2011.
[138] T. Hinrichs, N. Gude, M. Casado, J. Mitchell, and S. Shenker, "Expressing and enforcing flow based network security policies," in *Proc. ACM SIGPlan*, vol. 9, 2008.
[139] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, "Consistent updates for software-defined networks: Change you can believe in!" in *Proc. 10th ACM Workshop Hot Topics Netw.*, Nov. 2011, pp. 1–6.
[140] W. Han, H. Hu, and G. J. Ahn, "LPM: Layeblack policy management for software defined networks," in *Data and Applications Security and Privacy*. CA, USA: USENIX Association, 2014, pp. 356–363.
[141] F. A. Botelho, F. M. V. Ramos, D. Kreutz, and A. N. Bessani, "On the feasibility of a consistent and fault-tolerant data store for SDNs," in *Proc. 2nd Eur. Workshop Softw. Defined Netw.*, Oct. 2013, pp. 38–43.
[142] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation: A slice abstraction for software-defined networks," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Aug. 2012, pp. 79–84.
[143] R. W. Skowyra, A. Lapets, A. Bestavros, and A. Kfoury, "Verifiably-safe software-defined networks for CPS," in *Proc. 2nd ACM Int. Conf. High Confidence Networked Syst.*, Apr. 2013, pp. 101–110.
[144] A. Guha, M. Reitblatt, and N. Foster, "Machine-verified network controllers," *ACM SIGPLAN Notices*, vol. 48, no. 6, pp. 483–494, Jun. 2013.
[145] T. Nadeau and P. Pan. (2011). *Framework for Software Defined Networks*. Internet Draft. [Online]. Available: http://tools.ietf.org/id/draft-nadeau-sdn-framework-01.txt
[146] B. Isyaku, M. S. M. Zahid, M. B. Kamat, K. A. Bakar, and F. A. Ghaleb, "Software defined networking flow table management of OpenFlow switches performance and security challenges: A survey," *Future Internet*, vol. 12, no. 9, p. 147, Aug. 2020.
[147] O. Blial, M. B. Mamoun, and R. Benaini, "An overview on SDN architectures with multiple controllers," *J. Comput. Netw. Commun.*, vol. 2016, pp. 1–8, 2016.
[148] D. K. Ryait and M. Sharma, "Significance of controller in software defined networks," in *Proc. IEEE 15th Int. Conf. Ind. Inf. Syst. (ICIIS)*, Nov. 2020, pp. 561–566.
[149] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1339–1342, Aug. 2014.
[150] C. Yoon, S. Lee, H. Kang, T. Park, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Flow wars: Systemizing the attack surface and defenses in software-defined networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3514–3530, Dec. 2017.
[151] G. Logeswari, S. Bose, and T. Anitha, "An intrusion detection system for SDN using machine learning," *Intell. Autom. Soft Comput.*, vol. 35, no. 1, pp. 867–880, 2023.
[152] W. Li, Y. Wang, W. Meng, J. Li, and C. Su, "BlockCSDN: Towards blockchain-based collaborative intrusion detection in software defined networking," *IEICE Trans. Inf. Syst.*, vol. 105, no. 2, pp. 272–279, 2022.
[153] T. Mekki, I. Jabri, A. Rachedi, and L. Chaari, "Software-defined networking in vehicular networks: A survey," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 10, p. 4265, 2022.
[154] Y. Maleh, Y. Qasmaoui, K. El Gholami, Y. Sadqi, and S. Mounir, "A comprehensive survey on SDN security: Threats, mitigations, and future directions," *J. Reliable Intell. Environ.*, vol. 9, no. 2, pp. 201–239, Jun. 2023.
[155] A. Akhunzada, E. Ahmed, A. Ghani, M. K. Khan, M. Imran, and S. Guizani, "Securing software defined networks: Taxonomy, requirements, and open issues," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 36–44, Apr. 2015.
[156] A. Akhunzada, A. Gani, N. B. Anuar, A. Abdelaziz, M. K. Khan, A. Hayat, and S. U. Khan, "Secure and dependable software defined networks," *J. Netw. Comput. Appl.*, vol. 61, pp. 199–221, Feb. 2016.
[157] M. Safdar, Y. Abbas, W. Iqbal, M. Y. Umair, and A. Wakeel, "ARP overhead reduction framework for software defined data centers," *J. Netw. Syst. Manage.*, vol. 30, no. 3, p. 50, Jul. 2022.
[158] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10250–10276, Oct. 2020.

ARUSA KANWAL is currently a Lecturer with the Department of Information Security, National University of Sciences and Technology (NUST), Pakistan. Her research interests include but not limited to image processing, machine learning, information security, and secure systems.

MOHAMMAD NIZAMUDDIN is currently an Assistant Professor with the Cybersecurity Program, Department of Engineering, Physics, and Technology, Bronx Community College (BCC), City University of New York (CUNY). His research interests include cybersecurity, data science, machine learning, quantum computing, cloud computing, and algorithm development and time series analysis, with a specific focus on satellite data. He is actively involved in developing optical network architectures, conducting equipment testing, and implementing synchronization across diverse network infrastructures, particularly focusing on fiber optic equipment. In addition to his research pursuits, he contributes to BCC/CUNY's academic community by serving on both the curriculum committee and senate committee. With 22 years of teaching experience, he has imparted knowledge in electrical engineering and cybersecurity courses across various CUNY colleges.

WASEEM IQBAL received the bachelor's degree in computer sciences from the Department of Computer Science, University of Peshawar, in 2008, and the master's degree in information security and the Ph.D. degree from the National University of Sciences and Technology (NUST), Pakistan, in 2012. He is currently an Academician, a Researcher, a Security Professional and Industry Consultant. He has authored over 70 scientific research articles in prestigious international journals (ISI-indexed) and conferences. His professional services include, but not limited to industry consultation, a workshops organizer/resource person, a technical program committee member, a conference chief organizer, an invited speaker, and a reviewer of several international conferences. He is and has served as a guest editor for various journals.

WAQAS AMAN is currently an Assistant Professor with the Department of Information Systems, Sultan Qaboos University, Oman. His research interests include security in IoT-based smart critical infrastructure, security architecture and design, security and privacy concerns in big data, runtime ontologies, software security testing and assurance, information security education and awareness, adaptive security, and risk management.

YAWAR ABBAS is currently an Associate Professor with the Department of Computer Software Engineering, National University of Sciences and Technology (NUST). His research interests include software defined networking, software defined storage, AI, ML, wireless sensor networks, formal methods in software engineering, information security, cloud computing, data center networking, the IoT, security in SDN, WSN, and the smart IoT.

SHYNAR MUSSIRALIYEVA is currently the Head of the Information Systems and Cybersecurity Department, Al-Farabi Kazakh National University, Kazakhstan. Her research interests include machine learning, cybersecurity, and mathematical modeling.