International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2022/211595 Volume 9, Issue 1, January – February (2022)

SURVEY ARTICLE

Security Challenges and Related Solutions in Software Defined Networks: A Survey

Konda Srikar Goud
Department of Computer Science Engineering, GIT, GITAM (Deemed to be University), Visakhapatnam, Andhra Pradesh, BVRIT HYDERABAD College of Engineering for Women, Hyderabad, Telangana, India
kondasrikargoud@gmail.com

Srinivasa Rao Gidituri
Department of Computer Science Engineering, GIT, GITAM (Deemed to be University), Visakhapatnam, Andhra Pradesh, India
giduturisrinivasarao74@gmail.com

Received: 21 November 2021 / Revised: 31 December 2021 / Accepted: 03 January 2022 / Published: 28 February 2022

Abstract – In the current digitalized world, everything is interconnected and accessible from everywhere. Although traditional networks are widely adopted, their management is complicated. Therefore, they are not effective in providing services to the future Internet like a wide range of accessibility, high bandwidth, management, and security. On the other hand, Traditional network architecture relies on manual configurations of proprietary devices that are error-prone and inefficient to utilize the network devices properly. Software-defined Networking (SDN) has drawn massive changes in the traditional network paradigm by decoupling the network operations from the physical hardware and encouraging network control to be logically centralized. It provides network programmability and improves security by enabling a global view of the entire network and issues handled effectively by the centralized controller. As a result, SDN allows networks to monitor the traffic and detect vulnerabilities more effectively. It also simplifies the deployment of new services with more flexibility at a faster pace. On the other hand, the decoupling of control and the data planes introduces security threats such as Distributed Denial of Service (DDoS) attacks, Man in the Middle attacks, Saturation attacks, etc. As a result, SDN has attracted a lot of interest from both academics and industry. In this paper, we study security vulnerabilities on layers of SDN, the security frameworks that protect each layer, and many security methodologies for network-wide security.

Index Terms – Software Defined Networks (SDN), Open-Flow (OF), Network Operating System (NOS), Security, Reliability, Centralized Controller.

1. INTRODUCTION

Three layers represent computer Networking functionalities: Data plane, Control plane, and Application plane. The data plane comprises network devices that are capable of forwarding packets effectively [1]. The control plane handles interaction between the network-aware applications and the forwarding elements via different APIs. In particular, it translates application layer requirements to underlying forwarding devices, such as updating the forwarding table of switches or routers. Finally, the Application plane comprises applications that apply network control logic and techniques such as Simple Network Management Protocol (SNMP) service, which is responsible for remotely monitoring and configuring the control functionality. Some of the applications are Routing, Traffic Engineering, Firewalls, load balancing, etc. Therefore we can say that the Application plane creates network policies, the control plane executes them, and the data plane puts them into action by routing traffic properly.

In the legacy networks, the control plane and the data plane are firmly connected and incorporated in a single device [2], where the entire structure is decentralized, as shown in figure 1. Therefore, it brings a significant change in the Internet architecture in the early stages, which successfully achieved scalability, reliability, robustness, and high performance.

In the present world of digitalization, the legacy network architecture is not best suitable for the current increase in dynamic networking trends. Hence the management of networks is becoming more complex and challenging. It leads to the necessity of implementing complicated and high-level policies adaptable to current network environments and minimizing the burden on the low-level network devices. Moreover, in the present networking scenario, device miss-configuration-related issues are pretty standard, leading to unexpected network behavior. For instance, a single miss-configured device may compromise the entire network and makes it offline for hours.

There is a need for a novel approach is emphasized to solve numerous challenges related to legacy networks. In comparison with legacy network design, the primary purpose of the SDN paradigm is to simplify network operations, improve network administration and offer reliability, scalability, and security.

According to Kim and feamster [3], there are four essential factors in managing legacy networks.

- Complicated and minimal network configuration:
- Dynamic change in network topology
- The complexity of a decentralized and low-level network
- Heterogeneous proprietary devices

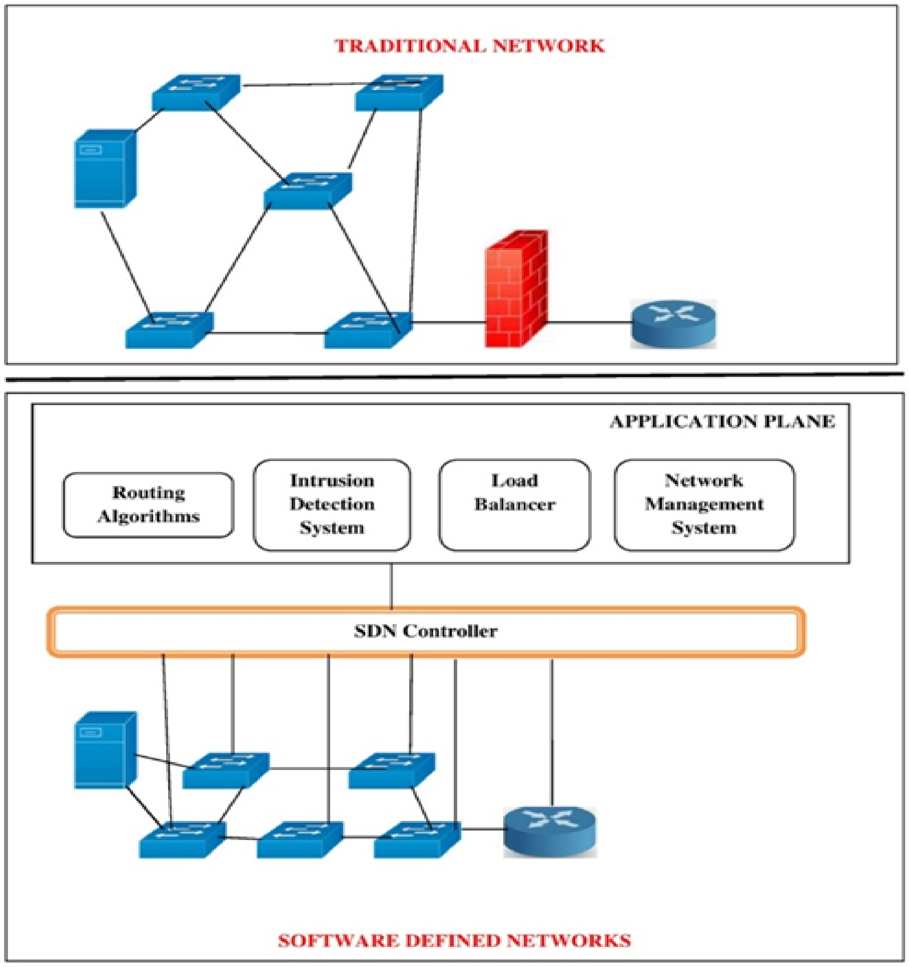

Figure 1 Traditional Networks (vs) Software Defined Networks

We can say that maintaining the network is complicated as the legacy network architecture is inadequate to deal with today's fast-paced network tendencies. SDN is one of the Open Network Foundations (ONF) initiatives, which introduced new open network architecture to address the challenges faced by the legacy networks. In traditional network architecture, the control plane and data plane are in a single unit. In contrast, the SDN separates the control and forwarding functionality by creating an abstraction between them [4].

As a result, all the network devices become essential forwarding elements. Furthermore, all the control plane functionality is moved to a logically centralized controller, making the network programmable and flexible in developing and deploying customized applications [1].

2. BACKGROUND AND RELATED WORK

Every device in the traditional network architecture has its data and control planes. The control plane of various devices interchanges topological information and builds its routing table that determines the path of the incoming packet to route through the data plane. In Software-Defined Networks (SDN), the control plane is separated from the underlying hardware and moved to a centralized entity called SDN Controller. As a result, network operators can have a centralized authority over network infrastructure. The data plane in the device forwards the data packets based on the flow table entries of the controller. For example, the flow entries consist of match fields and instructions.

When a new packet enters the switch, it checks for a particular match field in its flow table. If it is present, then the corresponding instruction will be executed. SDN is a new technology developed by the Open Networking Foundation (ONF) [5] that allows network administrators to design dynamic, cost-effective, flexible, secure, high-bandwidth, and adaptable to the latest developments. ONF is a non-profit organization that promotes network infrastructure and business models in developing open-source software and SDN protocols.

2.1. Advantages of SDN

Some of the advantages of SDN are:-

- Programmability: By decoupling the network services from the forwarding functionality, we can program and configure the network policies manually or using open source tools like OpenStack.
- Centralized management: Instead of configuring multiple devices, the network administrator can configure one centralized controller to distribute policies to all the connected switches.
- Agility and Flexibility: SDN's emergence helped many organizations quickly introduce new applications, operations, and hardware facilities to meet current business needs.
- SD-WAN: It is one of the SDN applications where it improves the operation and management of Wide Area Network by removing the control functionality from the underlying hardware. Therefore, network operators can remotely manage the networks.

2.2. SDN Challenges

As an advantage, SDN is also facing some challenges.

- Security Issues: An attacker can compromise different layers of SDN; therefore, a robust security solution is essential for additional layers of SDN.
- Controller Saturation attack: In the presence of only one control plane in the SDN environment, its computational resources get overwhelmed with massive traffic.
- Single Point of Failure: If the centralized controller gets compromised, the whole network will be affected.

2.3. SDN Architecture

The Software-Defined Network architecture is represented with three layers or planes, as shown in Figure 2.

- SDN Application layer/Plane
- SDN Control layer/Plane
- SDN Data or Infrastructure layer/Plane

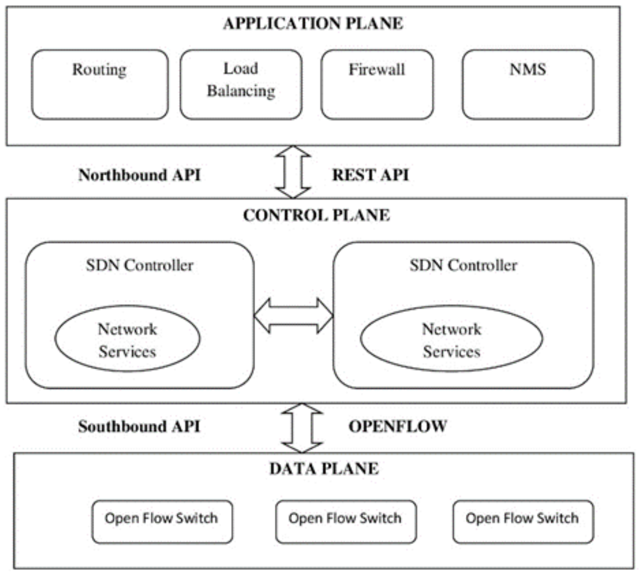

Figure 2 Layers of SDN Architecture

2.3.1. SDN Data Plane / Infrastructure Layer

The SDN Infrastructure layer comprises network devices like Switches, Routers, Firewalls, etc., the same as the legacy networks [1]. The significant distinction to the legacy physical devices is that these are just dumb forwarding devices without control logic to make routing decisions. Thus, the network intelligence is detached from the forwarding elements and shifted to a centralized controller. More crucially, SDN architecture is developed on top of the open standard API (i.e., Open Flow) effectively ensures configuration and communication across various control plane and data plane devices.

There are two primary entities in the SDN architecture, i.e., The Controller and the forwarding elements. The Forwarding device may be physical hardware or a logical entity that focuses on forwarding the packets. The controller is the “Brain of the Network” and also called Network Operating System, which runs on specialized hardware. The open flow enables switches to forward the packets based on flow tables. Each flow table entry consists of: a. Matching Rule, b. actions on a matched rule, c. counters for tracking matched packet statistics. When a new packet enters the open flow device, it checks the flow table entries in sequence. If a match finds in the entry, appropriate action performs. Generally, if there is no match, the most common scenario is to create and deploy a default rule in the switch such that the packet forwards to the controller. The flow rule is developed by considering various attributes like Source IP and Ethernet address, Destination IP and Ethernet address, source and destination port numbers, Packet Size, VLAN ID, VLAN priority, etc. [6]. If there is no match and no default rule created, then the switch drops the packet. Open flow enabled switches, routers, and other devices are available in the market and ready to deploy. The list of Open-flow devices is shown is table 1.

Switch Type	Switch Model	Make
Hardware	8200 zl & 5400zl [7]	Hawlett-Packard
	Arista 7150 Series [8]	Arista Networks
	Black Diamond X8 [9]	Extreme Networks
	CX600 Series [10]	Huawei
	EX9200 Ethernet [11]	Juniper
	MLX Series [12]	Brocade
	RackSwitch G8264 [13]	IBM
	Pica8 3920 [14]	Pica8
Software	Contrail-vRouter [15]	Juniper Networks
	LINC [16]	FlowForwarding
	ofSoftSwitch13 [17]	Ericsson
	OpenvSwitch [18,19]	Open Community
	Open-Flow Reference [20]	StanFord

Table 1 List of Open-Flow Devices

2.3.2. SDN Control Plane / Network Operating System

SDN isolates the control functionality from forwarding elements and places it in a centralized SDN controller. The SDN controller [21] implements the control plane functionalities. It manages and directs the entire network via the Network Operating System (NOS) from a centralized location. The NOS gathers data using APIs to analyze and operate a network similar to the operating system. As an essential component of SDN, the control plane is responsible for deploying flow configurations in the forwarding devices, i.e., Open-Flow switches. Based on the controller instructions, it performs all the flow operations.

The open flow protocol offers an open standard interface to interact with switches in SDN architecture [22]. When a new packet reaches the forwarding device, it searches the corresponding flow in its flow table. If a flow match exists, the instruction executes. Else, it forwards to the control plane. The list of the controllers is shown below in table 2.

Controller Name	Architecture	Programming Language
Beacon [23]	Centralised	Java
DISCO [24]	Distributed	Java
ElastiCon [25]	Distributed	Java
Fleet [26]	Distributed	-
FloodLight [27]	Centralised	Java
HP VAN SDN [28]	Distributed	Java
HyperFlow [29]	Distributed	C++
Kandoo [30]	Distributed	C, C++, Python
Onix [31]	Distributed	Python, C
Maestro [32]	Centralised	Java
Meridian [33]	Centralised	Java
NOX [34]	Centralised	C++
NVP Controller [35]	Distributed	-
OpenDayLight [36]	Distributed	Java
ONOS [37]	Distributed	Java
PANE [38]	Distributed	-
POX [39]	Centralised	Python
RoseMary [40]	Centralised	-
Ryu NOS [41]	Centralised	Python
SMaRtLight [42]	Distributed	Java
Trema [43]	Centralised	C, Ruby

Table 2 List of SDN Controllers

2.3.3. Application Layer/Plane

The Application layer comprises network applications and operations performed on the network, and we can implement a wide range of applications on this layer. For example, intrusion detection systems, firewalls, load balancers, network administration, network virtualization, and network monitoring are just a few available technologies. One of the main reasons for SDN adoption is the massive range of applications associated with real-world challenges. Besides their wide usage, we can classify most applications into five groups. Namely: Traffic Engineering, Security, Monitoring, Data Centre Networking. The list of the SDN applications is shown below in table 3.

Application Type	Application Name	Controller
Security	Active Security [44]	FloodLight
	AVANT-GUARD [45]	POX
	CloudWatcher [46]	NOX
	Cognition [47]	
	DDoS Detection [48]	NOX
	Elastic IP & security [49]	NOX
	Ethane [50]	-
	FlowNAC [49]	NOX
	FortNOX [50]	NOX
	FRESCO [51]	NOX
	LiveSec [52]	NOX
	Mapper [53]	-
	NetFuse [54]	-
	OpenSAFE [55]	NOX
	SANE [56]	SANE Controller
	VAVE [57]	NOX
Data Centre Networking	Big Data Apps [58]	-
	CloudNaaS [59]	NOX
	FlowComb [60]	NOX
	FlowDiff [61]	FlowVisor
	LIME [62]	FloodLight
	NetGraph [63]	-
	OpenTCP [64]	-
Monitoring	DCM [65]	DCM Controller
	OpenNetMon [66]	POX
	OpenTM [67]	NOX
	PaFlowMon [68]	FlowVisor
	PayLess [69]	FloodLight
Traffic Engineering	ALTO VPN [70]	NMS
	Aster*x [71]	NOX
	ElasticTree [72]	NOX
	MicroTE [73]	NOX
	Pronto [74]	Beacon
	QNOX [75]	NOX
	QueuePusher [76]	FloodLight

Table 3 List of SDN Applications

3. SECURITY RELATED OPEN CHALLENGES IN SDN ARCHITECTURE

SDN Layers	Attack Type	Description
Application	Authentication &Authorization	In many applications, there is an absence of proper Authentication and Authorization protocols which poses a significant risk. [78]
	Malicious Flow rule insertion	When an attacker deploys malicious applications, it generates a lot of malicious flow rules, and it will compromise the entire network, and it is hard to identify.
	Access control &Accessibility	Access control and Accountability are tough to implement on third-party applications.
Control	Distributed Denial of Service attack	Single point of control, Global view, centralized intelligence are the significant reasons for an attacker to implement DDoS on the Control plane
	Unauthorized access	There are no proper authentication protocols for the third-party applications while communicating with the control plane is the primary threat.
	Scalability	In the control pane, one

	&Availability	of the challenges is the centralized controller's scalability and availability issues.
Data plane	Malicious flow rules	As the switches are dumb, an attacker can insert malicious flow rules.
	DDos Attack	The switches can be overwhelmed by flooding malicious flows
	Compromising controller	When the controller gets compromised, the data plane will become dumb.
	TCP Attack	Transport Layer Security (TLS) is vulnerable and prone to this attacks
	Man in the Middle attack (MITM)	Most forwarding devices don't use TLS, leading to a MITM attack.

Table 4 Attacks on Layers of SDN

The overall network security depends on the safety of the controller. If the controller gets exploited, then the whole network goes down. A specific application needs to be deployed on the controller to mitigate threats like Spoofing, DoS, DDoS, MITM, Privilege escalation, insider threats, etc. A faulty Open-flow header can compromise the controller in a single controller environment, resulting in performance degradation and network services unavailable for legitimate users. Apart from the attacks mentioned above, there is a need for authentic protocols to validate and verify controllers in the distributed controller architecture. Those protocols enable authentication between controllers before sharing flow information. Because of two control planes and a wide range of switches, spoofing and DDoS are likely attacks in the hybrid controller environment. Hence, we need to protect SDN APIs with security protocols to mitigate from above risks.

Using the global perspective and centralized control of switches, an SDN controller can perform integrated network functionalities. On the other hand, the central authority of network control is more vulnerable to security risks. An attacker can easily compromise this single point rather than distributed environment [77]. When the attacker floods the controller with flow requests, its resources become overwhelmed and its services are unavailable to legitimate users. Therefore, we must authenticate the communication channel to overcome the above challenges before providing access to the control plane.

The usage of SDN in networking raised plenty of security issues. To get the full benefit from SDN, we need to address these issues by adopting security protocols. Here we are going to discuss attacks and threats on SDN layers. The list of various attacks on different layers of SDN is shown below in table 4.

3.1. Application Plane Security Challenges

Application authentication and authorization play a significant challenge in the present high-speed networks in the SDN environment. Network control and centralized intelligence are the two main characteristics that serve as the foundation for networking innovation. However, we implement many network functions as SDN apps. As a result, an illegitimate application can compromise the entire network, and its resources become unavailable to legitimate users. Here, we discuss the security challenges related to the application layer in detail.

The apps installed on the top of the application plane can impose severe security breaches and create a significant impact on the network's resources and operations. Even though Open-Flow allows us to install user-defined flow-based security apps, there are no predefined security apps installed and no standards to support open APIs for apps to control planes' resources and operations. Hence, an attacker can deploy the malicious app and can compromise the entire network. So, detecting a malicious application is an open challenge. Apart from that, a wide range of vendors and third-party application developers in the SDN open flow environment use various programming technologies, leading to interoperability, scalability, reliability, and security issues. Some of the challenges faced by the application plane are discussed below.

3.1.1. Threat From Third Party Applications

Third-party applications are responsible for the deployed applications on the controller. As a result, these applications gain administrative access and manipulate network resources and operations without appropriate security standards to safeguard the network from attackers.

Kreutz et al. [83] proposed three vectors to outline security threats in Software Defined Networks:

- Since there are no effective techniques for establishing a trustworthy connection between controller and applications, a malicious application can compromise the entire network.
- The attacker can compromise the application server and exploit the user credentials in the same way.

- Using these credentials, an attacker can infect malicious flows into the network.

There are several approaches for authenticating network resources, but there is no technique to authenticate applications in the SDN. Furthermore, the functionality of the network depends on the deployed applications. Therefore, there is necessary to develop a centralized approach to authenticate applications currently unavailable in the SDN environment. Hence it is one of the open challenges to the security of the application plane.

3.1.2. Threat Related to Access Management and Accountability

Access management and accountability are required to assure network security in SDN since applications are responsible for network operations. In this context, Hartman et al. [79] distinguished three types of applications. First, applications corresponding to network attributes like traffic path, traffic flow costs, routing, etc. Secondary applications are security related such as firewalls, IDS/IPS. The third types of application are Nested applications, which use the first two types of applications to perform their functionalities. As a result, a malicious application of the third type can use the second type of application and bypass the security and compromise the entire network and perform malicious activities. Hence access management and accountability are significant security challenges in the application plane.

The author in [80] demonstrates that the applications in the network can be SDN-based or non-SDN applications. SDN-based applications can directly communicate with the control plane, whereas non-SDN applications will use application datagrams for communication. In this context, an exploitable application might be a vehicle for illegal entry to the control plane. Hence access management among nested applications can be an open challenge in the application plane

3.2. Control Plane Security Challenges

As it is typical for the single controller to control all the heterogeneous devices in the network, we need to install additional controllers to achieve load balancing. It is challenging to deploy privacy policies for multiple controllers by dividing the network into sub-networks. The author in [81] developed and deployed an application on the SDN named ALTO (Application-Layer Traffic Optimization) to optimize network traffic. This application requires topological information from the controller. In this scenario, the malicious application access the network, leading to much security-related vulnerability.

The author in [82] stated that several controllers in a network might result in configuration issues because they are responsible for implementing network-wide services. In multiple controller environments, one of the significant challenges is synchronization issues. For example, if we make some changes in the network, all the controllers may not sync simultaneously. As a result, some apps like load balancers and firewalls may not work as they lack link-state change information in advance.

3.3. Data Plane Security Challenges

The control plane deploys the flow rules in the forwarding devices. Although the devices have limited TCAM, the flow rules have to deploy when a host sends the first packet to a new host or before a new host sends a packet. As the control logic decoupled from the switches, they cannot differentiate between malicious and legitimate incoming flow rules. Hence, the forwarding device's limited TCAM is the primary security challenge for the data plane to buffer the incoming flow requests until the controller deploys flow rules. It may also lead to DOS attacks [83].

The security of the data plane depends on the control plane’s security. When the centralized controller gets compromised, the entire network becomes unavailable, and the communication between the controller and switches will be lost. When there are no flow rules from the controller, the switches cannot forward the packets. Hence this will be another challenge to be focused. The attacker can take advantage of the decoupling nature of the switches and can deploy malicious flows or alter existing flow rules, leading to various attacks. It can be another security challenge for data planes.

The initial Open-Flow specifications include Transport Layer Security (TLS) and Datagram Transport Layer Security (DLS). However, the inclusion of TLS and DLS is not mandatory in the latest version of Open-Flow specifications. Hence, most vendors opted not to include TLS in their devices. It requires configuration overhead, which involves creating certificates and signing the certificates with cryptographic keys, and deploying the keys and certificates on all the devices. Therefore the absence of TLS can be vulnerable to open flow communication. According to Benton et al. [84], MITM attacks are severe in SDN networks compared to traditional networks because of the absence of TLS among the open flow communication. Hence hackers can gain complete access to the underlying devices and launch side-channel attacks. In addition to that, the author in [85] proved that unauthorized TCP communication could lead to many TCP level attacks in the Open-Flow network.

One of the open challenges in data plane security switches memory depletion due to fraudulent flows. When a new flow enters the device, it checks the relevant flow rule in the flow table. When a rule is present, it takes necessary actions, but it sends the flow to the corresponding control plane to get the rule if it is not available. This process may take some time. As an advantage, the attacker will flood such malicious flows to the switch so that the switch's TCAM will become overwhelmed and lead to a DDOS attack. In their paper [16], Dimitri et al. reported that restoration within less time in massive Open-Flow networks is complex. Furthermore, delay in installing flow rules in the Open-Flow switch might cause authentication and authorization problems and lead to side-channel attacks.

4. SOLUTIONS FOR THE SECURITY CHALLENGES IN SDN

The centralized controller can decide on the network's global view in SDN. In addition, the SDN framework inherits various security services such as monitoring and analyzing, which help in inserting new security policies and changing existing policies. It also supports a rapid and adaptable threat detection framework by capturing network intelligence and analyzing, updating, and reprogramming the network accordingly. SDN creates an environment where it is easy to design and deploy security policies in the forwarding devices. Therefore it eliminates the risk of device misconfigurations and policy conflicts in the networks. Furthermore, as SDN provides a global network view, it is easy to deploy security devices like firewalls and IDS/IPS according to the defined security policies. In this section, we will discuss various security solutions, strategies, Frameworks, platforms for protecting the SDN layers as shown in table 5.

SDN Layers	Proposed Solution	Solution Type
Application Layer	FRESCO [51]	To develop Open-Flow based security applications and Security framework
	PermOF [57]	Provides Authentication platform for Open-Flow Applications to access control and data plane.
	Assertion [88]	Developed an assertion based framework for validating and debugging SDN applications
	Flover [89]	The proposed control plane architecture will use a set of pre - defined parameters to evaluate flow rules given by the controller.
	OF-Testing	Proposed a framework for identifying malicious flows
Control Layer	SE-FloodLight	Developed Security enabled Floodlight controller, which provides an ideal secured control layer and secured Northbound API
	HybridCtrl	Developed a hybrid controller architecture works reactively to build routing and runs proactively to analyze traffic patterns
	DISCO [90]	Proposed a framework DISCO developed in collaboration with floodlight, which provides distributed and heterogeneous functionalities to control plane
	HyperFlow [91]	Proposed Hyperflow, an event-based control plane framework to increase the control plane scalability. It allows us to deploy various controllers which can able to make local decisions and decreases the flow setup time
	DDoS Detection	Proposed a DDoS detection framework which comprises of three modules and used SOM technique to discover hidden connections between the incoming flows into the network
Forwarding Layer	FortNOX [92]	Proposed a platform FortNOX, which allows NOX based controllers to analyze flow rule inconsistencies among Open-Flow applications
	FlowChecker [92]	Proposed a tool FlowChecker, which finds the errors in open flow rules in the forwarding devices

	VeriFlow [93]	Proposed a tool, VeriFlow, which helps in detecting and preventing malicious rules implemented by SDN applications
	Resonance	Proposed a tool Resonance, which uses real-time notifications and flowsstream information to implement dynamic access control policies to control traffic congestion in the network
	CPRecovery	Controller replicas are used in the proposed framework to keep the network functioning, even if some of the network portions gets compromised

Table 5 Proposed Solution for Various Challenges

4.1. Solutions for the Security Challenges in Application Plane

The control plane mediates the application and data planes in the SDN framework, hiding the network complexity. As a result, the centralized control plane provides a comfortable environment to deploy custom applications that access the network information and packet characteristics. Several programming languages, which include Frentic [94], Procera [95], and NetCore [96], are used to develop SDN applications.

The author in [51] proposed FRESCO. An efficient program to create new security applications that can deploy on any open-flow device. Moreover, they offered several security standards to determine whether SDN applications can act following security policies.

The author in [87] proposed PermOF an effective permission management system that allows Open-Flow applications to access control and data plane. A collection of privileges and isolation strategies is presented to implement permission management, including reading, writing, and notify. The application needs read permission to access the sensitive data and control the flow of sensitive data. We require notification permission to get notification messages to the applications when a specific event occurs. The write permission provides to control the application's ability to modify the behavior of data plane elements. Finally, the isolation strategy provides the controller an environment to administer the applications, separate control plane and data plane, and provide an interface to manage all the application's interactions with the outside world.

All the applications should maintain steady network's view and be adaptable to all the changes in the network. The author in [88] proposed a strategy for debugging and validating applications to achieve consistency and adaptability. Application developers can use the Assertion-based debugging tool to validate dynamic attributes of controller applications using high-level programs. Assertion techniques are helpful in the detection of errors in the code before the deployment of the applications. The strategy proposed in [88] is an algorithm that presents a mechanism to analyze the runtime environment's flow rules. The VeriFlow [93] verification algorithm is an incremental data structure that effectively validates the characteristics of all the changes in the network.

The author in [89] proposed a tool Flover. This Open-Flow-based application, deployed on the controller, verifies the flow rules generated by the control plane with the predefined rules. Hence, the flows don't deviate from the network security policies. The author also proposed other automatic testing tools to detect errors in Open-Flow programs. For example, the author in [97] proposed the ndb framework, which acts as a debugging tool for identifying the errors in the network programs. Furthermore, to track down and investigate the anomalies in the network, the author in [98] developed OFRewind. The ndb and the OFRewind frameworks effectively find malicious applications that can compromise the network.

4.2. Solutions for the Security Challenges in Control Plane

The solutions for the security challenges in the control plane depend on strategies and techniques used to secure from the following threats.

- Illegitimate or unauthorized applications
- Bypassing the security by focusing the control plane's scalability
- Distributed Denial of Service attacks
- Reliability issues in Controller placement

4.2.1. Malicious or Unauthorized Applications

As the applications require access to all network resources and data, it's essential to safeguard the control plane from malicious applications. Therefore it's the responsibility of the control plane to provide access to the legitimate applications in compliance with their security policies. The author in [99] proposed a secured floodlight controller (SE_Floodlight), which is an advanced version of the existing floodlight controller [100] which strives to secure the control plane. It integrates northbound API to the control plane, which acts as an interface between application and data plane, enabling privilege separation. Furthermore, it features an application authentication module for evaluating the integrity of the flow-rule generation module. The SE_FloodLight controller offers authorized privileges to applications, resolving rule conflicts by examining competing role generators' authority roles. It can also limit Packet_out messages generated by a variety of applications. Moreover, the SE_Floodlight controller has a new assessment module that can monitor any security-related activities in the control plane.

4.2.2. Bypassing the SDN Security by Targeting Control Plane’s Scalability

In the SDN environment, the controller deploys specific rules for every new client connection, resulting in many flows in the switches and creating a severe workload on the control plane. As a result, several strategies are proposed to reduce the burden on the controller. It also provides a wildcard strategy, allowing the controller to redirect the set of client requests to server replicas. To achieve high scalability and maintain load balancing on the control plane, it takes the leverage of wildcard rules of the forwarding devices.

The author in [101] compares re-active and pro-active open-flow controllers for scalability. Proactive controllers deploy rules in advance of the packet arriving at the switch, depending on specific preset rules. In contrast, the re-active controllers get the initial packet from the switch to update the flow table for that corresponding flow. According to the author in [101], the proactive controllers' scalability is likely more significant than the re-active controllers. Pro-active controllers require the traffic flows to be known ahead, which is not impossible in reality. As a result, the author proposes a hybrid controller strategy, wherein controllers work reactively to design routes while also possessing some intellect to proactively analyze the behavior of the flow and determine a route ahead of time.

Many initiatives are proceeding to improve the computational resources and distribute responsibilities among the controllers. For example, in [102], the author proposed McNettle, an enhanced control plane environment with high processing capability to process complex algorithms. Compared with NOX, McNettle is more scalable, scaling up to 46 cores versus ten cores for NOX and being more efficient in performance.

The author in [103], [90] proposed DISCO, a distributed controller which distributes the control plane's functionality. It implements the Advanced Messaging Queuing Protocol (AMPQ) [104], which is the enhanced version of the Floodlight controller [100]. The AMPQ is the collection of two modules: the intra-domain and the inter-domain modules. The intra-domain modules allow the controller to compute priority flow channels by monitoring the network and managing flow prioritization. In addition, these modules provide a dynamic solution to network problems by forwarding or dropping the traffic based on the severity of the packet. The inter-domain module consists of a messenger and agents, facilitating the interaction between controllers. The messenger detects adjacent controllers and establishes communication among them. The agents use the messenger's channels to communicate with other controllers on a network-wide level.

The author in [91] proposed HyperFlow, a scalable controller platform, which enables network programmers to install a set of controllers capable of making effective decisions to increase controller throughput and reduce latency. The author in [105], recommended that the controllers be placed at the starting point to minimize latency and then load balancing methods be used to distribute the load across the controllers.

4.2.3. Distributed Denial of Service Attacks

We can mitigate these attacks by analyzing the forwarding devices' incoming flow patterns and flow statistics. In open-flow, we can easily collect statistics of the switch from the Open-Flow controller. The author in [48] proposed a strategy by implementing Self Organizing Maps (SOM) [106]. SOM is a form of unsupervised neural network that converts an n-dimensional input pattern into a 1- or 2-dimensional map. We gather topological ordering and statistical properties for future analysis during the transformation process. The SOM procedure is adopted in [48] to uncover hidden relationships between packets arriving in the network.

The attack detection mechanism proposed in [48] contains three components: a flow_collector, a feature_extractor, and a classifier. The flow_collector collects flow from all of the flow tables of the forwarding devices at regular intervals. The feature extractor extracts critical features and sent to the classifier that helps in identifying an attack. Using SOM, the classifier determines if a particular tuple represents malicious or legitimate traffic. We have to train the SOM using a tremendous collection of samples captured during an attack or during regular traffic to produce a network topology with distinct areas representing different types of traffic. The trained SOM algorithm will distinguish traffic as either legitimate or malicious when we activate it.

4.2.4. Reliability Issues in the Controller Placement

The author in [105] highlights that; the scalability and robustness depend on the count of the controllers and topological design. These two are the significant challenges for the software-Defined network. As a result, researchers have given considerable importance to controller placement, and a broad range of algorithms have been investigated and implemented. For example, the authors in [107], [108], and [109] proposed the Simulated Annealing (SA) approach as the most effective strategy for placing the controller. In addition, a graph partitioning framework is developed in [110] to enhance robustness through effective controller placement.

The authors in [107] proposed a strategy where the controllers should synchronize and coordinate among themselves to improve the efficiency of SDN control plane operations. The procedures enhance the network by reducing the estimated loss percentage. The authors investigated various methodologies and their benefits for controller placement using realistic topologies.

The authors in [109] explore the Dynamic Controller Provisioning Problem (DCPP). It provides a support for the deployment of several controllers in a WAN environment. The size and positioning of controllers should update according to network conditions. DCPP analyzes traffic patterns to reduce the burden on the forwarding devices, proper coordination among the controllers, and the device to controller synchronization. Similarly, the author proposes a controller placement approach by providing a solid ratio between throughput and latency.

The authors in [111] propose a strategy to improve SDN resiliency. According to the author, one controller may not be sufficient to meet latency thresholds, but several controllers are required to meet network resilience requirements. Thus, the suggested strategy enhances the latency, throughput, resiliency, failure recovery and load balancing among the controllers.

4.3. Solutions for the Security challenges in Data plane

The illegitimate applications installed on the control plane can compromise the devices on the data plane. Furthermore, such programs can deploy, update, alter flow rules in the data plane as an initial point. Therefore, an efficient approach must authenticate and authorize the programs that modify the flow rules. FortNox [112] is a framework that allows the NOX Open-Flow controller to verify flow anomalies and authenticate the programs before altering the flow rules. FortNOX authorizes the flow rules using digital signatures before deploying on the data plane. It handles all flow rule insertion requests employing a live rule conflict detection and analysis engine. FortNOX prevents other apps from inserting contradictory flow rules in the network when a security application installs a flow rule.

FlowChecker [92] is an application that detects errors in Open-Flow rules within a switch or across multiple switches. To verify, analyze, and implement Open-Flow end-to-end configurations at runtime, FlowChecker is an Open-Flow application or a master controller. VeriFlow [93] is a network debugging tool for identifying and preventing malicious rules implemented by SDN applications from generating abnormal network activity. Even though a switch's performance depends on controller connectivity, the forwarding devices should have backup links. The Open-Flow devices employ connection detection techniques to verify controller connectivity, such as regularly transmitting activity probing packets to the controller. In addition, the Open-Flow protocol allows us to install a backup connection with a standby controller in case the primary controller fails. The authors in [113] recommend a controller replication strategy when the primary controller fails; it keeps the switch functioning. A switch regularly sends a probing signal to the control plane in this scenario. In addition, the switch assumes the controller is offline if it does not respond within a certain period. The Open-Flow switch then attempts to link with the redundant controller by executing a handshake and connecting instantly [113].

With effective network design and partitioning, we can improve the reliability of Open-Flow switches and their communication with controllers. An Open-Flow switch connected to the controller will be less vulnerable to saturation attacks because it will not be essential to keep unauthorized flows for more extended periods. The author in [114] evaluated that the increase in distance between the switch and the controller results in connectivity issues. As a result, the authors in [114] suggest that the distance between controllers and switches should be as minimal as possible to improve latency limitations. Still, it will also allow for faster restoration and maximize availability.

5. BENCH PERFORMANCE METRICS AND TOOLS

Here we discuss a collection of benchmarking metrics for SDN controllers. We use the keywords like Throughput and latency as basic metrics for performance measurement in SDN. We can also consider some other factors include security, reliability, availability, etc., to examine the performance of the SDN. The benchmark metrics can be categorized as follows.

- Throughput: The rate at which the controller processes the flow requests are generally referred to as Throughput. The speed at which the control plane receives packet_out messages from the open flow switch and generates corresponding flow at a unit time.
- Latency: Latency is the time the switch takes to transmit a packet to the controller and respond to the corresponding flow back to the switch. The Control plane’s latency depends on the control plane’s computational cost and delay.
- Flow installation based: This metric depends on the time taken to install the flow rules at all the switches in the network. Based on this metric, we can configure load balancing.
- System Performance: This metric accesses the control plane’s ability to use the SDN framework, hardware, Input/Output elements, etc., effectively. By implementing

multithreading, we can enhance the flow processing time of the control plane. Metrics such as power constraints, hardware failures, convergence time, link recovery, etc., can also be critical to evaluating SDN performance.

5.1. Benchmarking Tools

The below are the most often used benchmarking tools.

- CBench[115] is one of the famous benchmarking tools explicitly developed to evaluate the performance of the control plane. Throughput and latency are the primary assessment measures in CBench. We can calculate the Throughput by transmitting massive packet_in messages from each switch to estimate the controller's capacity. We can measure the latency by sending a packet_in request to the control plane and waiting for a reply. The above step repeats many times to calculate the average latency by the total number of responses received at each switch. This tool is limited to open flow 1.0 and 1.3 to access better performance of the SDN control plane.
- HCprobe is an enhancement to CBench developed using python to assess the performance of the control plane. Scalability and Reliability are the assessment measures in HCprobe. To test resilience and trustworthiness the simulated switch generates and transmits malicious packets to the control plane. As we run the test on the linux kernel we can acquire better control plane performance statistics.
- WCBench is a python-based updated version of CBench that automatically measures throughput and latency with graphical analysis. In addition, it supports a higher version of OpenFlow. But its support is confined to only particular versions of OpenDayLight Controllers.
- OFCBenchmark is a tool introduced to overcome the drawbacks of CBench. It consists of a Graphical User Interface (GUI) based console with a virtual switch and multiple clients to perform tests. In addition to existing benchmark metrics, it includes CPU consumption, rate of flow installation, Round trip time, etc., to access the performance of the SDN controller.
- OFCProbe[116] is an enhanced form of OFCBenchmark, which is platform-independent and focuses on improving the functionality and reducing the SDN control plane's overhead. It creates a large-scale simulation testbed with a substantial amount of switches and clients. This tool is intelligent in examining the network's impact in test conditions.
- OFNet[117] is a hybrid tool that provides an SDN simulation environment with control plane performance monitoring and debugging. In addition, it provides a large variety of network topologies and traffic generation modules to generate a variety of traffic patterns.

6. OVERVIEW OF RECENT RESEARCH RESULTS IN SDN SECURITY

Researchers initially used entropy to identify security threats in SDN due to its simplicity and low overhead. However, as research in SDN grows exponentially, researchers have claimed that entropy is inefficient in detecting some attacks, especially DDoS attacks, due to its high false positives rate. Therefore, mathematical and statistical models are widely adopted. Recently machine learning (ML) and deep learning (DL) have been extensively used to analyze network security threats. The author in [118] has concluded their results using ML algorithms. The Support Vector Machine has shown a significant performance compared with others. Its accuracy is about 97.50%, whereas the accuracy of Naive-Bayes is 96.03%, the Decision tree is 96.78%, the Logistic Regression is 89.98%. The author in [119] has concluded their results using DL algorithms. The accuracy of RNN is about 98.09%, whereas the accuracy of LSTM and GRU is 98.87% and 98.20%. But the worrying factor is its efficacy. The overall efficacy of the model depends on the dataset used to train the model. Unfortunately, most of the datasets used are obsolete with outdated attack patterns. As a result, these models cannot detect the latest attacks, which become a significant challenge. Recently, the security of the SDN has enhanced with the evolution and adoption of blockchain technology. However, despite multiple advantages, we still face several security threats due to the limitations in SDN design, high processing time, and computational cost, affecting the network's confidentiality, integrity, and availability.

7. CONCLUSION AND FUTURE WORKS

With the gradual introduction of SDN architecture, new threats are pretty likely to evolve. Similarly, the attack surface is likely to expand because standard network security risks spread alongside SDN-related security challenges. On the other hand, SDN aims to bring innovation to communication networks. According to the previous studies on SDN security, it enables the rapid development of cost-effective security services. To ensure security, various security policies are enabled in the network, and the security demands of multiple applications, services, resources, and devices are collected by the distribution layer of SDN. The network programmability and centralized control can improve the reliability and scalability of the networks.

Moreover, SDN significantly improves security by achieving a global view of the entire network. But on the other side, new security challenges are evolving. This paper has discussed various open challenges related to security in SDN's application, control, and data planes. Also, we have presented some of the flaws related to security and techniques to strengthen the network and mitigate attacks related to security in SDN. . In this work, we have found most of the existing SDN controllers lack security protocols and standards. Therefore, we can conclude that the existing SDN architecture must be strengthened, upgraded, and enhanced to address the aforementioned challenges.The open challenges which were discussed in section 3 are the future directions for the research community.

REFERENCES

[1] Singh, Jagdeep and Sunny Behal. “Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions.” Comput. Sci. Rev. 37 (2020): 100279.

[2] Fonseca, Paulo César, and Edjard Souza Mota. "A survey on fault management in software-defined networks." IEEE Communications Surveys & Tutorials 19, no. 4 (2017): 2284-2321

[3] Kreutz, Diego, Fernando MV Ramos, and Paulo Verissimo. "Towards secure and dependable software-defined networks." In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 55-60. 2013.

[4] Jo, Hyeonseong, Jaehyun Nam, and Seungwon Shin. "Nosarmor: Building a secure network operating system." Security and Communication Networks 2018 (2018).

[5] Open Network Foundation: achievements, https://opennetworking.org/about-onf/careers/about-onf/ Accessed 04 Jan 2021

[6] Kandoi, Rajat, and Markku Antikainen. "Denial-of-service attacks in OpenFlow SDN networks." In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 1322-1326. IEEE, 2015.

[7] HP, ‘‘8200 ZL switch series,’’ 2013. [Online]. Available: http://h17007.www1.hp.com/ us/en/networking/products/switches/HP_8200_zl_Switch_Series/.

[8] Arista Networks, ‘‘7150 Series,’’ 2013. [Online]. Available: http://www.Aristanetworks.com/media/system/pdf/Datasheets/7150S_ Datasheet.pdf.

[9] Extreme Networks, ‘‘Blackdiamond x8,’’2013. [Online]. Available: http://www.extremenetworks.com/libraries/products/ DSBDX_1832.pdf.

[10] Huawei Technologies Co., Ltd., ‘‘Cx600 metro services platform,’’ 2013. [Online].Available: http://www.huawei.com/ucmf/groups/public/documents/attachments/hw_132369.pdf.

[11] Juniper Networks, ‘‘Ex9200 Ethernet switch,’’ 2013. [Online]. Available: http://www.juniper.net/us/en/local/pdf/datasheets/1000432-en.pdf.

[12] BROCADE, ‘‘MLX Series,’’ 2013. [Online].available: http://www.brocade.com/ products/all/routers/product-details/ netiron-mlx-series/system-options.page

[13] IBM, ‘‘System networking RackSwitch G8264,’’ 2013. [Online]. Available: http://www-03.ibm.com/systems/networking/switches/rack/g8264/

[14] Pica8, ‘‘3920,’’ 2013. [Online]. Available: http://www.pica8.org/documents/pica8-datasheet-64x10gbe-p3780-p3920.pdf.

[15] .Juniper Networks, Inc., ‘‘Contrail virtual router,’’ 2013. [Online]. Available: https:// github.com/Juniper/contrail-vrouter.

[16] Rutka, Krzysztof, Konrad Kaplita, Sandhya Narayan, and Stuart Bailey. "LINC Switch (2013)."

[17] Fernandes, Eder Leao, and Christian Esteve Rothenberg. "OpenFlow 1.3 software switch." Salao de Ferramentas do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos SBRC (2014): 1021-1028.

[18] Open vSwitch, 2013. [Online]. Available: http://vswitch.org/

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

[20] Open-Flow Community, ‘‘Switching reference system,’’ 2009. [Online]. Available: http:// www.Open-Flow.org/wp/downloads/

[21] Xie, Haiyong, Tina Tsou, Diego R. Lopez, en Hongtao Yin. “Use Cases for ALTO with Software Defined Networks”. Internet Engineering Task Force, 27 Junie 2012. https://datatracker.ietf.org/doc/html/draft-xie-alto-sdn-use-cases-01.

[22] McKeown, Nick, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM computer communication review 38, no. 2 (2008): 69-74.

[23] Erickson, David. "The beacon openflow controller." In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 13-18. 2013.

[24] Phemius, Kevin, Mathieu Bouet, and Jérémie Leguay. "Disco: Distributed multi-domain sdn controllers." In 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1-4. IEEE, 2014.

[25] Dixit, Advait, Fang Hao, Sarit Mukherjee, T. V. Lakshman, and Ramana Rao Kompella. "ElastiCon; an elastic distributed SDN controller." In 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), pp. 17-27. IEEE, 2014.

[26] Matsumoto, Stephanos, Samuel Hitz, and Adrian Perrig. "Fleet: Defending SDNs from malicious administrators." In Proceedings of the third workshop on Hot topics in software defined networking, pp. 103-108. 2014.

[27] Floodlight, P. "Project floodlight open source software for building softwaredefined networks." (2012).

[28] HP, ‘‘SDN controller architecture,’’Tech. Rep., Sep. 2013.

[29] Tootoonchian, Amin, and Yashar Ganjali. "Hyperflow: A distributed control plane for openflow." In Proceedings of the 2010 internet network management conference on Research on enterprise networking, vol. 3. 2010.

[30] Hassas Yeganeh, Soheil, and Yashar Ganjali. "Kandoo: a framework for efficient and scalable offloading of control applications." In Proceedings of the first workshop on Hot topics in software defined networks, pp. 19-24. 2012.

[31] Koponen, Teemu, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan et al. "Onix: A distributed control platform for large-scale production networks." In OSDI, vol. 10, pp. 1-6. 2010.

[32] Cai, Z., A. L. Cox, and T. S. E. Ng. "Maestro: A System for Scalable OpenFlow Control. Technical report." Rice University (2011).

[33] Banikazemi, Mohammad, David Olshefski, Anees Shaikh, John Tracey, and Guohui Wang. "Meridian: an SDN platform for cloud network services." IEEE Communications Magazine 51, no. 2 (2013): 120-127.

[34] Gude, Natasha, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. "NOX: towards an operating system for networks." ACM SIGCOMM computer communication review 38, no. 3 (2008): 105-110.

[35] Koponen, Teemu, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev et al. "Network virtualization in multi-tenant datacenters." In 11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14), pp. 203-216. 2014.

[36] OpenDaylight, A. "Linux Foundation Collaborative Project." Dispontvel online: http:/Avww. opendaylight. org (2013).

[37] U. Krishnaswamy et al., ‘‘ONOS: An open source distributed SDN OS,’’ 2013. [Online]. Available: http://www.slideshare.net/ umeshkrishnaswamy/open-networkoperating- system.

[38] Ferguson, Andrew D., Arjun Guha, Chen Liang, Rodrigo Fonseca, and Shriram Krishnamurthi. "Participatory networking: An API for

application control of SDNs." ACM SIGCOMM computer communication review 43, no. 4 (2013): 327-338.

[39] McCauley, Murphy. "About pox." URL: http://www. noxrepo. org/pox/about-pox/. Online (2013).

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

[41] Telegraph, Nippon. "Telephone Corporation,"Ryu Network Operating System."." (2012).

[42] Botelho, Fábio, Alysson Bessani, Fernando MV Ramos, and Paulo Ferreira. "On the design of practical fault-tolerant SDN controllers." In 2014 third European workshop on software defined networks, pp. 73-78. IEEE, 2014.

[43] Takamiya, Yasuhito, and Nick Karanatsios. "Trema OpenFlow controller framework." (2012).

[44] Hand, Ryan, Michael Ton, and Eric Keller. "Active security." In Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, pp. 1-7. 2013.

[45] Shin, Seungwon, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. "Avant-guard: Scalable and vigilant switch flow management in software-defined networks." In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 413-424. 2013.

[46] Shin, Seungwon, and Guofei Gu. "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)." In 2012 20th IEEE international conference on network protocols (ICNP), pp. 1-6. IEEE, 2012.

[47] Tantar, Emilia, Maria Rita Palattella, Tigran Avanesov, Miroslaw Kantor, and Thomas Engel. "Cognition: A tool for reinforcing security in software defined networks." In EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V, pp. 61-78. Springer, Cham, 2014.

[48] Braga, Rodrigo, Edjard Mota, and Alexandre Passito. "Lightweight DDoS flooding attack detection using NOX/OpenFlow." In IEEE Local Computer Network Conference, pp. 408-415. IEEE, 2010.

[49] Stabler, Greg, Aaron Rosen, Sebastien Goasguen, and Kuang-Ching Wang. "Elastic IP and security groups implementation using OpenFlow." In Proceedings of the 6th international workshop on Virtualization Technologies in Distributed Computing Date, pp. 53-60. 2012.

[50] Casado, Martin, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. "Ethane: Taking control of the enterprise." ACM SIGCOMM computer communication review 37, no. 4 (2007): 1-12.

[51] Shin, Seung Won, Phillip Porras, Vinod Yegneswara, Martin Fong, Guofei Gu, and Mabry Tyson. "Fresco: Modular composable security services for software-defined networks." In 20th Annual Network & Distributed System Security Symposium. Ndss, 2013.

[52] Wang, Kai, Yaxuan Qi, Baohua Yang, Yibo Xue, and Jun Li. "LiveSec: Towards effective security management in large-scale production networks." In 2012 32nd International Conference on Distributed Computing Systems Workshops, pp. 451-460. IEEE, 2012.

[53] Sapio, Amedeo, Mario Baldi, Yong Liao, Gyan Ranjan, Fulvio Risso, Alok Tongaonkar, Ruben Torres, and Antonio Nucci. "MAPPER: a mobile application personal policy enforcement router for enterprise networks." In 2014 Third European Workshop on Software Defined Networks, pp. 131-132. IEEE, 2014.

[54] Wang, Ye, Yueping Zhang, Vishal Singh, Cristian Lumezanu, and Guofei Jiang. "Netfuse: Short-circuiting traffic surges in the cloud." In 2013 IEEE international conference on communications (ICC), pp. 3514-3518. IEEE, 2013.

[55] Ballard, Jeffrey R., Ian Rae, and Aditya Akella. "Extensible and Scalable Network Monitoring Using OpenSAFE." Inm/wren 10 (2010).

[56] Casado, Martin, Tal Garfinkel, Aditya Akella, Michael J. Freedman, Dan Boneh, Nick McKeown, and Scott Shenker. "SANE: A Protection Architecture for Enterprise Networks." In USENIX Security Symposium, vol. 49, p. 50. 2006.

[57] Yao, Guang, Jun Bi, and Peiyao Xiao. "Source address validation solution with OpenFlow/NOX architecture." In 2011 19Th IEEE international conference on network protocols, pp. 7-12. IEEE, 2011.

[58] Wang, Guohui, TS Eugene Ng, and Anees Shaikh. "Programming your network at run-time for big data applications." In Proceedings of the first workshop on Hot topics in software defined networks, pp. 103-108. 2012.

[59] Benson, Theophilus, Aditya Akella, Anees Shaikh, and Sambit Sahu. "CloudNaaS: a cloud networking platform for enterprise applications." In Proceedings of the 2nd ACM Symposium on Cloud Computing, pp. 1-13. 2011.

[60] Das, Anupam, Cristian Lumezanu, Yueping Zhang, Vishal Singh, Guofei Jiang, and Curtis Yu. "Transparent and flexible network management for big data processing in the cloud." In 5th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 13). 2013.

[61] Arefin, Ahsan, Vishal K. Singh, Guofei Jiang, Yueping Zhang, and Cristian Lumezanu. "Diagnosing data center behavior flow by flow." In 2013 IEEE 33rd International Conference on Distributed Computing Systems, pp. 11-20. IEEE, 2013.

[62] Keller, Eric, Soudeh Ghorbani, Matt Caesar, and Jennifer Rexford. "Live migration of an entire network (and its hosts)." In Proceedings of the 11th ACM Workshop on Hot Topics in Networks, pp. 109-114. 2012.

[63] Raghavendra, Ramya, Jorge Lobo, and Kang-Won Lee. "Dynamic graph query primitives for sdn-based cloudnetwork management." In Proceedings of the first workshop on Hot topics in software defined networks, pp. 97-102. 2012.

[64] Ghobadi, Monia, and Y. Ganjali. "TCP adaptation framework in data centers." PhD diss., University of Toronto, 2013.

[65] Yu, Ye, Chen Qian, and Xin Li. "Distributed and collaborative traffic monitoring in software defined networks." In Proceedings of the third workshop on Hot topics in software defined networking, pp. 85-90. 2014.

[66] Van Adrichem, Niels LM, Christian Doerr, and Fernando A. Kuipers. "Opennetmon: Network monitoring in openflow software-defined networks." In 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1-8. IEEE, 2014.

[67] Tootoonchian, Amin, Monia Ghobadi, and Yashar Ganjali. "OpenTM: traffic matrix estimator for OpenFlow networks." In International Conference on Passive and Active Network Measurement, pp. 201-210. Springer, Berlin, Heidelberg, 2010.

[68] Argyropoulos, Christos, Dimitrios Kalogeras, Georgios Androulidakis, and Vasilis Maglaris. "PaFloMon--A Slice Aware Passive Flow Monitoring Framework for OpenFlow Enabled Experimental Facilities." In 2012 European Workshop on Software Defined Networking, pp. 97-102. IEEE, 2012.

[69] Chowdhury, Shihabur Rahman, Md Faizul Bari, Reaz Ahmed, and Raouf Boutaba. "Payless: A low cost network monitoring framework for software defined networks." In 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1-9. IEEE, 2014.

[70] Scharf, Michael, Vijay Gurbani, Thomas Voith, Manuel Stein, W. Roome, Greg Soprovich, and Volker Hilt. "Dynamic VPN optimization by ALTO guidance." In 2013 second European workshop on software defined networks, pp. 13-18. IEEE, 2013.

[71] Handigol, Nikhil, Srini Seetharaman, Mario Flajslik, Aaron Gember, Nick McKeown, Guru Parulkar, Aditya Akella et al. "Aster* x: Load-balancing web traffic over wide-area networks." Open Networking Summit Demo (2011).

[72] Heller, Brandon, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown.

"Elastictree: Saving energy in data center networks." In Nsdi, vol. 10, pp. 249-264. 2010.
[73] Benson, Theophilus, Ashok Anand, Aditya Akella, and Ming Zhang. "MicroTE: Fine grained traffic engineering for data centers." In Proceedings of the seventh conference on emerging networking experiments and technologies, pp. 1-12. 2011.
[74] Xiong, Pengcheng, and Hakan Hacigümüş. "Pronto: A software-defined networking based system for performance management of analytical queries on distributed data stores." Proceedings of the VLDB Endowment 7, no. 13 (2014): 1661-1664.
[75] Jeong, Kwangtae, Jinwook Kim, and Young-Tak Kim. "QoS-aware network operating system for software defined networking with generalized OpenFlows." In 2012 IEEE Network Operations and Management Symposium, pp. 1167-1174. IEEE, 2012.
[76] Palma, David, Joao Goncalves, Bruno Sousa, Luis Cordeiro, Paulo Simoes, Sachin Sharma, and Dimitri Staessens. "The queuepusher: Enabling queue management in openflow." In 2014 third European workshop on software defined networks, pp. 125-126. IEEE, 2014.
[77] Ahmad, Suhail, and Ajaz Hussain Mir. "Scalability, consistency, reliability and security in sdn controllers: A survey of diverse sdn controllers." Journal of Network and Systems Management 29, no. 1 (2021): 1-59.
[78] Park, Younghee, Hongxin Hu, Xiaohong Yuan, and Hongda Li. "Enhancing Security Education Through Designing SDN Security Labs in CloudLab." In Proceedings of the 49th ACM Technical Symposium on Computer Science Education, pp. 185-190. 2018.
[79] Kreutz, Diego, Fernando MV Ramos, and Paulo Verissimo. "Towards secure and dependable software-defined networks." In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 55-60. 2013.
[80] Xie, Haiyong, Tina Tsou, Diego R. Lopez, en Hongtao Yin. "Use Cases for ALTO with Software Defined Networks". Internet Engineering Task Force, 27 Junie 2012. https://datatracker.ietf.org/doc/html/draft-xie-alto-sdn-use-cases-01.
[81] Seedorf, Jan, and Eric Burger. Application-layer traffic optimization (ALTO) problem statement. RFC 5693, October, 2009..
[82] MR, Harshitha, Harshitha JS, Brunda KS, and Shrihari MR. "An Approach for Supervising the Security Threats using Software Defined Networks." Available at SSRN 3510055 (2019).
[83] Gupta, Brij B., Gregorio Martinez Perez, Dharma P. Agrawal, and Deepak Gupta. Handbook of computer networks and cyber security. Springer, 2020.
[84] Benton, Kevin, L. Jean Camp, and Chris Small. "OpenFlow vulnerability assessment." In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 151-152. 2013.
[85] Liyanage, Madhusanka, and Andrei Gurtov. "Secured VPN models for LTE backhaul networks." In 2012 IEEE Vehicular Technology Conference (VTC Fall), pp. 1-5. IEEE, 2012.
[86] Staessens, Dimitri, Sachin Sharma, Didier Colle, Mario Pickavet, and Piet Demeester. "Software defined networking: Meeting carrier grade requirements." In 2011 18th IEEE workshop on local & metropolitan area networks (LANMAN), pp. 1-6. IEEE, 2011.
[87] Wen, Xitao, Yan Chen, Chengchen Hu, Chao Shi, and Yi Wang. "Towards a secure controller platform for openflow applications." In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 171-172. 2013.
[88] Beckett, Ryan, Xuan Kelvin Zou, Shuyuan Zhang, Sharad Malik, Jennifer Rexford, and David Walker. "An assertion language for debugging SDN applications." In Proceedings of the third workshop on Hot topics in software defined networking, pp. 91-96. 2014.
[89] Canini, Marco, Dejan Kostic, Jennifer Rexford, and Daniele Venzano. "Automating the testing of OpenFlow applications." In The 1st International Workshop on Rigorous Protocol Engineering (WRiPE). 2011.
[90] Phemius, Kévin, Mathieu Bouet, and Jérémie Leguay. "DISCO: Distributed SDN controllers in a multi-domain environment." In 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1-2. IEEE, 2014.
[91] Tootoonchian, Amin, and Yashar Ganjali. "Hyperflow: A distributed control plane for openflow." In Proceedings of the 2010 internet network management conference on Research on enterprise networking, vol. 3. 2010.
[92] Al-Shaer, Ehab, and Saeed Al-Haj. "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures." In Proceedings of the 3rd ACM workshop on Assurable and usable security configuration, pp. 37-44. 2010.
[93] Khurshid, Ahmed, Xuan Zou, Wenxuan Zhou, Matthew Caesar, and P. Brighten Godfrey. "Veriflow: Verifying network-wide invariants in real time." In 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13), pp. 15-27. 2013.
[94] Foster, Nate, Rob Harrison, Michael J. Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. "Frenetic: A network programming language." ACM Sigplan Notices 46, no. 9 (2011): 279-291.
[95] Voellmy, Andreas, Hyojoon Kim, and Nick Feamster. "Procera: a language for high-level reactive network control." In Proceedings of the first workshop on Hot topics in software defined networks, pp. 43-48. 2012.
[96] Monsanto, Christopher, Nate Foster, Rob Harrison, and David Walker. "A compiler and run-time system for network programming languages." Acm sigplan notices 47, no. 1 (2012): 217-230.
[97] Handigol, Nikhil, Brandon Heller, Vimalkumar Jeyakumar, David Maziéres, and Nick McKeown. "Where is the debugger for my software-defined network?." In Proceedings of the first workshop on Hot topics in software defined networks, pp. 55-60. 2012.
[98] Wundsam, Andreas, Dan Levin, Srini Seetharaman, and Anja Feldmann. "OFRewind: Enabling record and replay troubleshooting for networks." In USENIX Annual Technical Conference, pp. 327-340. USENIX Association, 2011.
[99] "Porras, Phillip A., Steven Cheung, Martin W. Fong, Keith Skinner, and Vinod Yegneswaran."Securing the software defined network control layer." In NDSS. 2015.
[100] Switch, Big. "Developing floodlight modules. Floodlight OpenFlow controller." (2012).
[101] Fernandez, Marcial P. "Comparing openflow controller paradigms scalability: Reactive and proactive." In 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pp. 1009-1016. IEEE, 2013.
[102] Voellmy, Andreas, and Junchang Wang. "Scalable software defined network controllers." In Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, pp. 289-290. 2012.
[103] Phemius, Kevin, Mathieu Bouet, and Jérémie Leguay. "Disco: Distributed multi-domain sdn controllers." In 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1-4. IEEE, 2014.
[104] Vinoski, Steve. "Advanced message queuing protocol." IEEE Internet Computing 10, no. 6 (2006): 87-89.
[105] Heller, Brandon, Rob Sherwood, and Nick McKeown. "The controller placement problem." ACM SIGCOMM Computer Communication Review 42, no. 4 (2012): 473-478.
[106] Kohonen, Teuvo. "The self-organizing map." Proceedings of the IEEE 78, no. 9 (1990): 1464-1480.
[107] Hu, Yannan, Wendong Wang, Xiangyang Gong, Xirong Que, and Shiduan Cheng. "On reliability-optimized controller placement for software-defined networks." China Communications 11, no. 2 (2014): 38-54.
[108] Fan, Yuqi, and Tao Ouyang. "Reliability-aware controller placements in software defined networks." In 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 2133-2140. IEEE, 2019.

[109] Bari, Md Faizul, Arup Raton Roy, Shihabur Rahman Chowdhury, Qi Zhang, Mohamed Faten Zhani, Reaz Ahmed, and Raouf Boutaba. "Dynamic controller provisioning in software defined networks." In Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), pp. 18-25. IEEE, 2013.
[110] Zhang, Ying, Neda Beheshti, and Mallik Tatipamula. "On resilience of split-architecture networks." In 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011, pp. 1-6. IEEE, 2011.
[111] Hock, David, Matthias Hartmann, Steffen Gebert, Michael Jarschel, Thomas Zinner, and Phuoc Tran-Gia. "Pareto-optimal resilient controller placement in SDN-based core networks." In Proceedings of the 2013 25th International Teletraffic Congress (ITC), pp. 1-9. IEEE, 2013.
[112] Porras, Philip, Seungwon Shin, Vinod Yegneswaran, Martin Fong, Mabry Tyson, and Guofei Gu. "A security enforcement kernel for OpenFlow networks." In Proceedings of the first workshop on Hot topics in software defined networks, pp. 121-126. 2012.
[113] Fonseca, Paulo, Ricardo Bennesby, Edjard Mota, and Alexandre Passito. "A replication component for resilient OpenFlow-based networking." In 2012 IEEE Network operations and management symposium, pp. 933-939. IEEE, 2012.
[114] Seedorf, Jan, and Eric Burger. Application-layer traffic optimization (ALTO) problem statement. RFC 5693, October, 2009.
[115] Sherwood, Rob, and K. K. Yap. "Cbench controller benchmarker." Last accessed, Nov (2011).
[116] Jarschel, Michael, Christopher Metter, Thomas Zinner, Steffen Gebert, and Phuoc Tran-Gia. "OFCProbe: A platform-independent tool for OpenFlow controller analysis." In 2014 IEEE Fifth International Conference on Communications and Electronics (ICCE), pp. 182-187. IEEE, 2014.
[117] Shankar, Ganesh H. "OFNet." OFNet-Quick User Guide.[Online]. Available: http://sdninsights. org/.[Accessed: 05-Jun-2018] (2016).
[118] Ahmad, Ahnaf, Erkki Harjula, Mika Ylianttila, en Ijaz Ahmad. “Evaluation of Machine Learning Techniques for Security in SDN”. In 2020 IEEE Globecom Workshops (GC Wkshps, 1–6, 2020. https://doi.org/10.1109/GCWkshps50303.2020.9367477.
[119] Alshra’a, Abdullah Soliman, Ahmad Farhat, and Jochen Seitz. "Deep Learning Algorithms for Detecting Denial of Service Attacks in Software-Defined Networks." Procedia Computer Science 191 (2021): 254-263.

Authors

Konda Srikar Goud received his M.Tech degree from Computer Science and Engineering from JNTU Hyderabad. He is working as Assistant Professor in the Department of Information Technology, BVRIT Hyderabad sCollege of Engineering for Women, Hyderabad, India. He is currently pursuing Ph.D. at GITAM University, Visakhapatnam, Andhra Pradesh, India. Research area includes Network Security, computer networks and Software Defined Networks.

Srinivasa Rao Giduturi is Associate Professor in the department of Computer Science and Engineering, GITAM University, Visakhapatnam, Andhra Pradesh, India. He received his M.Tech degree from Anna University; Ph.D. degree from GITAM University. Research interest includes Mobile Computing, Computer Networks.

How to cite this article:

Konda Srikar Goud, Srinivasa Rao Gidituri, “Security Challenges and Related Solutions in Software Defined Networks: A Survey”, International Journal of Computer Networks and Applications (IJCNA), 9(1), PP: 22-37, 2022, DOI: 10.22247/ijcna/2022/211595.