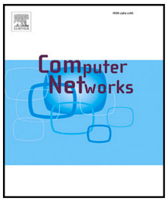
Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Review article

A comprehensive survey of vulnerability and information security in SDN

Raktim Deb [*], Sudipta Roy

Department of Computer Science and Engineering, Triguna Sen School of Technology, Assam University, Silchar, Assam 788011, India

ARTICLE INFO

Keywords:
Software defined network
Component vulnerability
SDN information security and threats

ABSTRACT

SDN changes the networking vision with an impressive thought of segregating the networking control from the data management hardware and brings new functionalities such as programmability, elasticity, flexibility, and adoption capability in the network, which are difficult to think of in traditional rigid network architecture. However, a wide range of vulnerable surfaces directly or indirectly affect the SDN-based system's information security and launch various attacks. The paper begins with a glimpse of the advantages of SDN over the traditional network but, the findings of the research work take off the wraps regarding vulnerabilities and their consequences on information security. Consequently, the threat surfaces are exposed that exist in SDN architecture due to weak information security. In addition, the research findings also disclose other prominent issues irrespective of information security issues. The inclusion intends to ring the bell in the maximum SDN aspects and make researchers or professionals aware of current trends of SDN in the best possible way. The comprehensiveness of this work is retained by detailing every part of SDN, which helps the researchers or professionals to improve SDN structurally or functionally.

1. Introduction

With the advent of the high demand network service era for different users, the network service provider requires a highly flexible network architecture and network components to support network communications flexibility. The Software Defined Network (SDN) comes with the thought of separating the networking control Operating System (OS) from the hardware functionality and plotting the control OS in a centralized position to manage the underneath hardware functionalities. Not only that, SDN provides a standardized Application Programming Interface (API) for adding new programmable features in the networks at any period that overcome the problem of lack of programmability and flexibility of traditional network architecture. In other words, the benefit of these properties of SDN is that it helps the network service provider to achieve a network architecture that is more open, flexible, programmable, and manageable towards fulfilling the high demand network services [1]. These advantageous properties help SDN to achieve the global view of underneath topology and empower the control plane to dynamic modification of network functionalities and the abstractions from hardware concerns. Although these properties reconstruct network architecture and network communication management advantageous compared to the traditional network but on the down-side, it also brings new challenges to the implementation of security in the SDN. For example, the separation of the functionalities of control and data plane, limitation of the flow table in the network component makes the door open for Denial of Service (DoS) or network adversary attack. Due to the lack of explicit practices to open programmability, various functions (data plane and control plane) and components are under severe SDN trust management threat.

This paper intends to identify the information security issues in SDN and provides a comprehensive survey regarding the vulnerability and information security threats of SDN along with addressing the enhanced information security due to the introduction of SDN architecture and the individual vulnerabilities of each component and function of SDN, causing new security threat management challenges. Successful addressing of these mentioned issues will help researchers, industry peoples, and beginners to identify SDN architecture and its working procedure, its blessings, and the consequences of new challenges in the networking environments.

The rest of the paper is structured as follows: Section 2 demonstrates an overview of our motivation. Section 3 provides a short trip to SDN generation, an abstract view of SDN network architecture and its communication. We have also discussed some network functionality and how SDN improves its concern in the networking system. Sections 4, 5, and 6 represent chronological order to expose information security risk in SDN architecture. Each section has an impact on other sections so that Section 4 addresses the SDN vulnerabilities for which information security issues occur in the system, like global view and open programmability lead to a vulnerable surface for authentication

[*] Corresponding author.
E-mail address: debraktim@gmail.com (R. Deb).

https://doi.org/10.1016/j.comnet.2022.108802
Received 10 September 2021; Received in revised form 11 January 2022; Accepted 18 January 2022
Available online 5 February 2022

and authorization issues. Section 5 addresses these issues elaborately. By exploiting these issues (individually or jointly), a malicious user may generate several threats or attacks in different layers of SDN architecture. Section 6 illustrates possible attacks in different SDN layers. The survey work undergoes deep observations of several existing survey papers and recent or popular existing research works to pinpoint maximum possible issues that SDN architecture or SDN-based networking faces to date. Once the observation is completed (root cause, impacts, consequences), the observation outcomes are distributed in three sections (Sections 4, 5, and 6). Accordingly, references are placed in the respective sections, which are the most recent or most popular research, and more importantly, justify the utterance of that particular section or the issue addressed in the section. Each section also includes a comparative statement among the related references for a particular issue mentioned in that section. The inclusion of the references in such a manner is that the researchers can easily grab the knowledge about the current trends of a particular issue of the respective sections and proceed to future improvement. Vulnerable components are not the only issue in the SDN. There are a few other issues addressed in Section 7 for which SDN suffers and restricts its wide adoption. Section 8 visualizes a few advanced research directions that might reduce the existing information security issues convincingly. Finally, we conclude by pointing out the future dimensions of this evolving networking area in Section 9.

2. Motivation

To secure any network system, the first step is to identify the flaws. To implement protection in the network from unwanted damages, we must have ideas about network entities and communication transactions. To implement network security, a networking system must have the following properties: confidentiality, integrity, availability, authentication, and non-repudiation [2]. The alterations introduced by the SDN in the network architecture should sustain all these properties to ensure network security. It is worthy to mention that information security means auditing the five network parameters, as mentioned above. In the networking domain, there exist two perceptions regarding auditing mechanisms: (1) Network provider: audit the network to identify the pitfalls of network and make network informationally secure, (2) Intruder/Attacker: identify the pitfalls in information security and attack the vulnerable components. Therefore, the SDN domain requires an illustrative study regarding information security such that considering the illustration, researchers can propose a more secure environment. There is no research study available highlighting these issues and set our goal to convey the present study.

D. Kreutz et al. performed an advanced analysis in SDN architecture. They have identified new vulnerabilities specific to SDN due to the nature of federated network control and providing the programmability to the network [3]. In a recent vulnerability study, the authors in [4] observe different control planes, mainly three Java-based control planes (POX, Floodlight, OpenDaylight (ODL)), identify the basic sets of pitfalls, and design weakness creates the vulnerable surface for the SDN control planes. In comparison, the present study tries to concentrate on information security depending on SDN standard specifications. The discussion is not limited to any programming domain-specific control plane. Any SDN control plane, either existing one or proposed in the future, must follow the same SDN standard specifications. Depending on the pitfalls and design weaknesses of three control planes, the authors [4] observe 12 vulnerabilities over SDN and then categorizes those into three information security parameters, i.e., confidentiality, integrity, and availability. Whereas the present study chronologically explores vulnerabilities in SDN standard specification and how this impacts information security. Lastly, it addresses the attack surfaces or information security threats due to vulnerability impacts over information security. Moreover, the present study is not limited to discuss three information security parameters only; but extended to all the information security parameters, i.e., confidentiality, authenticity, integrity, consistency, and availability.

In another recent survey [5], the authors present two different views of SDN security. In one hand, the authors demonstrate “how SDN basic features improve security in SDN”. In this section, the authors define 7 existing threat vectors and 14 attack surfaces. Although authors convincingly demonstrate all the aspects, it is identified that SDN poses more than 12 threat vectors, and more importantly, this survey work is unable to produce granular observation such as “why that threat vector exits in SDN domain”, “how do they impacting information security of SDN” and “what are the consequences of last two questions in SDN architecture?” In order to produce granular observation, the present study investigates SDN specification and addresses the root cause of component-wise vulnerabilities and their impacts on information security, and, lastly addresses the attack surface by addressing the threat vectors. Thus, the present study investigates different manners and produces more granular observation than the existing one [5]. The outcome of the present study identifies 22 attacks surfaces in different layers of SDN architecture and addresses the corresponding root cause and threat vectors.

On the other hand, the authors in [5] address how SDN basic features improve security in SDN, but that is not the only case SDN can do. The SDN features undoubtedly increase the SDN security but also improve many networking services as well. The present study concisely addresses each of these parameters. In addition, it demonstrates the downside of SDN features in other networking services also. Therefore, the present study comprehensively produces a granular observation and enhanced information regarding SDN aspects compared to existing ones.

Although the OpenFlow specification specifies the adoption of Transport Layer Security (TLS) [6] with a cooperative authentication mechanism, the lack of standardization of TLS reduces wide adoption in the SDN architecture. R. Klöti et al. have performed a security analysis of OpenFlow protocol where the main concern is to identify and mitigate information disclosure and DoS attack by implementing STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation), and attack tree modeling method [7].

These demonstrations give a clear idea about the range of information security issues in the SDN architecture. This study investigates several survey papers where authors concentrate on architectural functionalities, components of SDN, and briefing the loopholes of SDN so that researchers can identify the future direction in SDN. For instance, [8,9] represent a comprehensive survey about state-of-art programmable networks, and [10] elaborate layered taxonomy of SDN architecture.

The research works [11–14] deal with a brief discussion in security and threats in SDN. These research studies mainly focus on categorizing the attacks for which SDN architecture is susceptible and demonstrate possible countermeasures within 2015. In a recent work [15], undergo a survey to identify existing flaws of SDN based cloud computing architecture. This research work especially highlights the flaws that cause Distributed Deniel of Service (DDOS) attacks in the respective environment. Similarly, Mathebula et al. [16], escalate the survey in the Software-Defined Wireless Sensor Networks(SDWSN) domain. This survey work illustrates the security challenges, specifically in the SDWSN domain and also its existing solutions. That means by the passing year, SDN spreads its wings and leaves a new vulnerable surface that increases the possibilities of new kinds of attacks such as Drown attack, Reverse Loop, Topology Freezing, Link Latency Inspect (LLI), and many others discussed in Section 6. However, the existing surveys mainly concentrate on SDN security, and accordingly, discuss associated information security such as [11]. Whereas the present study concentrates on finding out vulnerabilities lie in each SDN entity, their consequences on SDN information security that escalate the attack surface of SDN and point out different kinds of attacks generated by

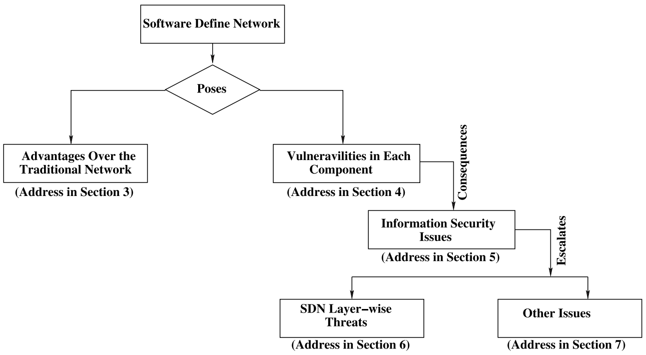


Fig. 1. Flow diagram of the paper.

misleading information security and hidden component vulnerabilities. The right hand side of Fig. 1 depicts the same.

The study profoundly goes through the research works carried out during 2005–2020, where during the year 2017–2020, researchers disclose few new vulnerabilities and attacks over the SDN due to existing limitations of information security in the SDN domain. Thus, global identification of such new problems, existing solutions, and future direction are required. Whereas none of the previous surveys, as referred in this paper, address those issues.

The previous surveys either concentrating on describing the architectural functionalities and corresponding pros and cons of SDN [8,9] or illustrating the possible attack in SDN architecture [11–14]. However, no such illustration is present that ring the bell in all aspect of SDN and aware the researchers regarding of all these aspects in a single go.

That motivates us to do the research study. The actual intention of this paper is to address information security issues and their consequences over an SDN-based environment. However, to reach our motivation, contribution begins with a concise discussion of SDN and its enhanced properties and moves forward by disclosing the vulnerabilities and their corresponding effect on information security. After then, it illustrates the information security and the threat surface it creates for attack generation. Lastly, a depiction of a few other issues irrespective of information security in SDN architecture is also incorporated. Fig. 1 depicts the complete flow of contributions of this paper.

The inclusion of a concise discussion of SDN and the other matters helps to enlighten the existing positivity and negativity of SDN. Thus, this work tries to retain the comprehensiveness of the survey work, and more importantly, our motivation by detailing SDN in a maximum aspect. This survey work will help beginners, and the researcher to understand SDN in each small part with a single go and opens the door to improve SDN structurally or functionally. The next section provides a quick view of SDN architecture and communication procedures.

3. Overview

The separation of network computing capabilities is a long dream for the network professionals as the tightly coupled control plane, and data plane significantly increases complexity and cost of the network management. Authors M. Caesar et al. encouraged this segregation in the first place in 2005. They had managed IP forwarding component separation from a centralized routing management system for dealing Intra, and Inter routing in the network [17]. Soon after, many researchers accelerate this segregating concept and introduce a 4D project (4 planes:decision, dissemination, discovery, and data) [18], Secure Architecture for the Networked Enterprise (SANE) [19], Ethane [20] but, among them, Ethane can be considered as the parental representation of present generation SDN. Ethane is the first proposition that brought fine-grain simple-to-use network policy implementation during communications and transformed switch procedure into a table. However, it suffers from broadcast and service discovery, application routing, agreeing with non-standard ports, and spoofing Ethernet. Ethane's idea and limitation set the building blocks of today's SDN standard by proposing the OpenFlow protocol. OpenFlow protocol is an open standard that specifies the functionalities of switches named as OpenFlow (OF) switch, communication between control-data plane, and a centralized operating system that manages OpenFlow switch is called controller or control plane.

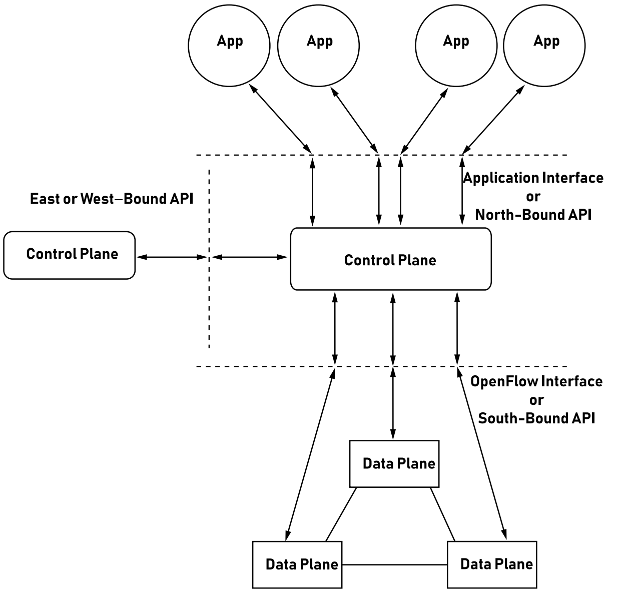


Fig. 2. Conceptual architecture of SDN.

3.1. What is SDN?

Fundamentally, the SDN comes with the perception of segregating the control mechanism from the forwarding devices, dump devices that can only solve packet processing. An intellect control plane manages upper and lower-level networking components. The research works [8,9] represent a comprehensive survey about state-of-art programmable networks, and [10] elaborate layered taxonomy of SDN architecture. These extensive studies identified that conceptual SDN architecture ornamented with three APIs: northbound, southbound, and east/westbound API, besides fundamental architecture adds more functionalities as shown in Fig. 2. The southbound API is a conceptually standardized representation of OpenFlow protocol for managing control-data plane communications and underneath topology in the SDN architecture. The northbound API interacts with network applications but, no standardization is mentioned for the northbound one as it needs interaction with vendor-agnostic applications in the network. The east/westbound is a conditional API that comes in the act only when multiple control planes interact during network communications.

3.2. Why SDN?

The surveys [8–10] also include architectural and communication procedural differences between SDN and traditional network. However, SDN passes more than 15 years of its existence, and identifying these differences is no longer enough. We have undergone several research contributions where researchers prioritize SDN architecture for implementation over the traditional architecture of different area domain and extract the key reasons to differentiate SDN and traditional architecture in contrast to (1) Network Management, (2) Network Security, (3) Quality of Service (QoS), (4) Quality of Experience (QoE), and (5) Cost Reduction.

3.2.1. Enhanced network management

Network management is a noticeable issue while dealing with the dynamic demand for resources is increased. The network management: integration, segregation, reconfiguration, automation of the network communications, is simple in SDN due to its root characteristics of programmability, flexibility [8–10]. The network administrator/manager only needs to make changes in the control program from the centralized point of the architecture.

3.2.2. Enhanced network security

Network security is another big issue and the most attractive area in the computing environment for researchers and intruders. The researchers keep themselves busy to provide much more strict security imposition in the network while intruders keep busy themselves to bypass those implications. Consistent firewall management is a tedious task in traditional networks, whereas SDN can auto shape the traffic. Imposing firewall in SDN is quite easy as the network manager/administrator needs to introduce an application on top of the control plane. OF switches can act as a filter to direct the flow to the control plane as in the network. Thus, real-time detection is easier in SDN environment [11–14]. Besides, Bring-Your-Device security is also easily manageable in an SDN environment where the administrator can impose additional functionality in the control plane [21,22].

3.2.3. Enhanced quality of service

The traditional network experiences difficulty in providing QoS due to non-standardization and required hop to hop management for vendor-specific firmware in each varying component whereas, SDN can provide higher-level policy abstraction without intervening underneath topology in fewer time [23,24]. Therefore, the control plane can execute a different algorithm that administrates QoS flow rules in the switch. Thus, other network architectures explicitly adopt SDN functionality for improving QoS in their domain. Feng et al. [25] and Manzanares-Lopez et al. [26] adopt SDN functionalities to improve QoS in IEEE 802.11 networks. Network virtualization is one of the key technologies for the abstracting network to increase service delivery. This abstracting mechanism is tedious in the traditional network due to hop management, whereas SDN with Network Virtualization Function (NVF) is much more efficient in producing granular services. A. Hammad demonstrates that SDN and NVF are complementary to each other to improve network service [27]. Thus, the discussion demonstrates that SDN has more capability than traditional to provide QoS.

3.2.4. Enhanced quality of experience

The QoE is subjective to network resources' execution capability in utility service to answer user expectations concerning their current state. The rigidity of traditional networks makes it difficult to reach such expectations whereas, in SDN, global view, programmable north & southbound interface, and ability to dynamic decision making in per-flow rule installation make it easier to cope with user expectation [28]. A.B Letifa implements a machine learning algorithm over SDN architecture and shows that SDN architecture outperforms traditional network to provide QoE in contrast of End to End delay and throughput [29]. Not only that, later in their extended work, they have identified Mean Opinion Score (MOS) to make a comparison between SDN and traditional networks in contrast to video streaming services where it shows similar advantages [30]. The above discussion is fair enough to justify that SDN can outperform traditional networking in contrast to QoE.

3.2.5. Enhanced scalability

The SDN is emerged due to the limitation of traditional network in which network resources and scalability functionality is a long-term issue. Whereas SDN can provide intellectual service as the ability of programmability, automated flow management allows a network administrator to set up highly scalable, flexible, and rapidly adaptable to varying business needs architecture [31]. The Bera et al. [32], Shif et al. [33], Romero Gázquez and Delgado [34] and Tayyaba et al. [35] represent extensive surveys in Internet of Things (IoT) over SDN network. The observation of these surveys' outcomes identified that the IoT environment reduces seamless services due to traditional architecture's rigidness. The SDN functionality is a prominent solution to achieve seamless services in the IoT environment. Google is a prominent cloud provider that recognizes the SDN capability of enhancing scalability in network and service management. Therefore, they have modified their cloud and data centers with the SDN architectural component underneath in place of traditional architectural components [36,37]. Thus SDN can outperform the traditional network in contrast to enhancing scalability in the network.

3.2.6. Network function virtualization

SDN architecture and Network Function Virtualization (NFV) architecture represent network abstraction, but both the architectures perform the network abstraction differently, for different purposes. The NFV covers the virtualization of networking functions like MME, firewall, IDS, DNS whereas, SDN forwards the data packet from one network to another network and also provides the networking function for routing. The policy definitions are building the path and pipelines where network traffic can reach to the destination. However, both the technologies are complementary to each other in order to achieve a complete networking solution.

The NFV helps in the virtualization to build hardware and building blocks ready for networking services i.e., new services or new nodes can be ready for a network at no point in time but, it is unable to establish connectivity between the resources. The production of dynamic network functions is not worthy as long as there is no connectivity between them. The connectivity requires multiple things like IP allocation, bandwidth allocation, policy opening, routing changes, end-to-reachability, and more importantly, those things are required as fast as NFV provides the network functions. At this point, SDN plays a key role and is capable of providing all such connectivity dynamically and also as fast as NFV creates resources. Therefore, SDN and NFV jointly produce a fast, dynamic, secure, and ready-to-use network efficiently to the networking society. Many advanced technologies such as cloud computing use SDN-NFV to enhance their security. The authors J-L Luo et al. implement SDN and NFV to create a security service function tree (SecSFT) [38]. The SecSFT classifies suspicious flows from the mixed network flow. To classify the network flow, each node of SecSFT implements decision rules using virtualized security-related network functions (Example: load balancing, and traffic shaping), and virtualized network security functions (e.g., intrusion detection, firewall). The SDN and NFV prominently increase the security in the IoT environment. The research work by Zarca et al. [39] shows significant use of SDN-NFV to achieve IoT security. This research work introduces a semantic-aware, zero-touch, and policy-driven security orchestration framework that is able to produce optimum allocation of service function chaining in an IoT environment. The authors P Krishnan et al. also introduce Distributed Threat Analytics and Response System (DTARS) in fog/edge computing environment by implementing SDN-NFV [40]. The DTARS can detect real-life DDoS/botnets such as Mirai with a 95% accuracy rate.

3.2.7. Cost

The cost reduction modeling intends to identify the network architecture that can optimize the network's service cost. The SDN features create an attractive environment for the industry people and researchers to consider its financial return to the service providers. In this respect, N. Zhang et al. equate Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) in Long-Term Evolution (LTE) networks with and without SDN architecture. The observed results identify that SDN architecture as a base architecture can reduce CAPEX up to 7% per annum [41]. Moreover, centralized control and programmable management of dynamic aggregation and segregation can reduce over-provisioning and under-provisioning network problems.

Therefore, in the scenario where mobile operators share the same network resources, the SDN can substantially reduce CAPEX up to 58.04% [42]. The OPEX mainly depends on hardware and software utilization in the network. The SDN with NVF bring (1) automated elasticity, (2) software–hardware independence, (3) hardware uniformity, (4) virtualized software infrastructure, and (5) multi-tenancy and resource polling; that helps to optimize the OPEX [43]. By realizing that HIS Technology Team generates a census report by accompanying interviews of those service providers who have implemented or planned to implement SDN [44]. The census observation indicates that 85% of respondents expect the OPEX cost reduction up to 50% if the core component of the network architecture is SDN. Therefore, the network infrastructure with SDN as core architecture incurs less cost compared to traditional networking.

3.3. Section conclusion

This section introduces the positive intent of SDN in contrast to network parameters. SDN is a new environment and still evolving. It also extracts out many points from the drawn references to justify the stand. It does not include implementations as those are not very significant to this survey's actual intention. The survey papers [11–14] illustrated security aspects based on the layered architecture of SDN, but this paper intended to generalize vulnerabilities of the components in the first place and gradually distinguishing security risk and threats caused by the risk. The next section demonstrates such vulnerabilities.

4. Vulnerabilities in SDN

As mentioned in the above section, the SDN properties those eliminate the traditional network's deficiencies create new loopholes in the SDN and reduce the SDN's wide adoption. Besides SDN architectural components such as OpenFlow switches, the control plane and end-host are vulnerable to zero-day vulnerabilities [45]. The statement can be justified by S. Lee et al.'s research work that introduced an assessment framework to reveal the unknown vulnerabilities of SDN architecture [46]. The framework implements fuzzy techniques and ex-post-facto analysis to inspect vulnerabilities — the ex-post-facto outlines seven test criteria for detecting SDN architecture vulnerabilities. The fuzzification mechanism over test criteria verifies 20 known attack scenarios and observes seven new attack scenarios resulting from the misleading of the control plane application. The observation depends on some Java based SDN control planes such as Open Network Operating System (ONOS), OpenDaylight (ODL), Floodlight, and Brocade Vyatta- a commercial control plane. The research work does not provide any generic mechanism to test in other control planes, and also, it is not an automatized mechanism; it requires human effort.

The research work creates positive intent in our research goal and justifies that SDN vulnerabilities are uncovered and under exploration. Therefore, getting proper knowledge of those vulnerabilities is an essential task for beginners and the research industry so that special care can be taken in the first place while implementing the SDN.

4.1. Control plane vulnerability

Segregating the control plane from the data plane and making it a core component for managing the whole network from a single point makes SDN vulnerable. If an intruder somehow grabs the control of a centralized control plane, then he/she can cause devastating consequences in the networking system. The control plane or precisely control plane operating system provides a global view of the network to the applications running on top. The global view of SDN acts as a two-edged sword for the SDN environment. On one side, it provides the simplest way for the network programmer to manage the network activities, and on the other hand, it opens the door for a malicious user to do unfair activities. OpenFlow specification does not mention a standardized security mechanism for control plane applications. As a reason, a malicious application might be able to run on top of the control plane OS and can modify the network behavior or grab the network control with or without the intervention of intruder effort. Nonetheless, an application could have vulnerabilities that can create an open door for malicious activity in the network.

The programmable feature provides an environment where different software developers can contribute their codes or software to the programmable control plane. There is the possibility of those codes to contain some errors or vulnerabilities. An intruder can target such vulnerabilities and grabs the opportunity to compromise the network or do some malicious activity. In this respect, the authors J. Sonchack et al. propose a security application capable of installing applicationspecific processing and switch programme to do the monitoring task [47]. It provides an interface to each control plane application for which an application dynamically installs or invokes the corresponding security control mechanism. Therefore, any misleading activities are restricted near to data plane and make network resources available to the authentic applications. There are many issues regarding the proposition that need consideration: (1) It requires an optimization mechanism for the interface between the control plane and the data plane. (2) Verifying the OFX using other popular control plane security applications. (3) The proposed mechanism implemented using Ryu and OpenFlow switch pica 8. It requires to test the same with other variations of control plane and OpenFlow switches. (4) The implementation depends on the C and Python programming environment.

The control plane suffers from a weak authentication mechanism and an entrusted relationship between the control plane and applications running on top of it due to the non-existence of a standardized security mechanism in OpenFlow specification. The weak authentication mechanism opens the door for unauthorized access to the network properties without an intruder's intervention. In this respect, authors K. K. Karmakar et al. introduce a policy-based security application capable of defending and managing network behavior during network communications [45,48]. It introduces an Evaluation Engine to evaluate the incoming traffic flow according to appropriate security policies. Topology repository maintains underneath topology information. Enforcer enforces policies in the data plane for the underneath topologies.

For this validation, it modifies the OpenFlow protocol by incorporating a hash-based rule matching mechanism. The security is responsible for maintaining coordination between a switch and a northbound security application. However, Karmakar et al. [45] are limited in inter domain verification, whereas [48] cover inter and intra domain policy management. Our previous works [49,50] mathematically verify that the control plane is the most vulnerable component in SDN and visualizes that vulnerability increases when other SDN components are mutually vulnerable by implementing a Bayesian network.

A single point control plane suffers from resource saturation or data plane to control plane communication link saturation due to incompetence of scaling the resources and extensive communication between data and control plane under high traffic load in usual conditions. This incompetence degrades the network's performance and fails to reach the user expectation [31]. Multiple control-plane environments can reduce such incompetence of SDN [32]. In such architecture, control planes can share their state and the current status of resources. If a failure occurs in the control plane or control plane resources, then another control plane can take the previous one and run network activities normally and increase the network's resiliency.

Transport Layer Security in the control plane is another solution to reduce resource saturation. However, with these circumstances control plane fails to authenticate the switches in the control plane side and leaves the network vulnerable. For such vulnerability, an intruder can identify the network status by observing the traffic flow and can disclose the network information. Addressing this authentication issue authors, J. W. Kang et al. introduce a new control plane and switch design named "Mynah" [57]. They used OpenFlow vendor

Table 1
Summary of control plane vulnerability in existing research works.

Vulnerability	Proposed solution	Implementation	Mechanism	Future work
1. Centralized single point control or global view of the network has become a two-edged sword for SDN architecture. It might expose the underneath topology of a system. 2. Open Programmability contributes to an environment that increases the possibilities of erroneous code entries. 3. There is no such access control mechanism of code entries. 4. There is no such access control mechanism for control plane application. 5. There exists weak authentication among the control plane and control plane application communication. 6. There is no or weak authentication among control plane and data plane communication. 7. Control plane and data plane communication resources are rigid. 8. Communication mechanism depends on the plaintext messages. 9. Communication channel is not synchronous. 10. There is no availability of standardized security mechanisms or TLS implementation.	1. Fine-grained policy-based enforcement [45].	1. It uses the POX control plane.	1. Security application at the control plane specifies access policies, and a security agent in the northbound interface enforces the access rules in the data plane. This mechanism covers vulnerability 1.	1. Inter-domain policy management.
	2. SDN Intra and Inter-domain communication management.	2. ONOS control plane used as a core architecture [48].	2. Policy and Topology repository, Policy manager, and enforcer are four different control plane modules communicates to manage the access. It covers vulnerability 1 and 2.	2. –
	3. Role-based authentication [51,52].	3. Floodlight control plane [51], NOX control plane [52].	3. Network traffic management policies introduced to manage granular services [51,52]. It overcomes vulnerability 3.	3. The control plane kernel-level requires granular security.
	4. Access control mechanism [53,54].	4. Works on the control plane.	4. The Klaedtke et al. [53] extends the Unix operating system as a control plane, and The Monaco et al. [54] is proposition only. This mechanism covers vulnerability 4.	4. Both require implementation in the large scale practical domain.
	5. Increase security among control plane and application communication [55].	5. Blockchain-based Authentication–Authorization-Accounting System (BlockAS) is introduced between control plane and application communication.	5. An application generates an access request to BlockAS, and the BlockAs verify the request and generate a token with a new status. The network admin verifies the token and issues it for the accession of resources. This mechanism covers vulnerability 5.	5. Performance needs to improve for handling requests and increase the concurrent requests time.
	6. Multiple control-plane architectures can reduce the rigidity issue.	6. Architectural design [31,32].	6. Literature survey. The survey for improving vulnerability 7.	6. –
	7. Blockchain and Attribute-based encryption are used to communicate between apps and the multi-control domain [56].	7. SDN-IoT app ledger (SILedger) is introduced and distributed in the multiple heterogeneous networks.	7. SILedger manages decentralized access control using ABE and blockchain. The application authorization tokens are encrypted by ABE. Only the precise attributes of an application can decrypt the token and be able to access the resources. It can improve vulnerability 7 and 8.	7. This mechanism is for the SDN-IoT environment and might be implemented in other domains. Accession delay during decryption can be improved.
	8. Control plane to data plane authentication using Data plane Identifiers (DPID).	8. Works on a combination of data plane and control plane [57].	8. The switch consists of an encryption mechanism and control plane perform the DPID based authentication for network traffic. It can improve vulnerability 6, 9 and 10.	8. Inclusion of TLS/SSL.
Risk in control plane	Confidentiality, Authenticity, Integrity, Consistency, Availability			

extensions to implement Mynah. It introduces lightweight Data Plane Identifiers (DPID) based authentication mechanism distributed in the control plane and the data plane. Mynah's outcome compared with the general SDN component is that it is a lightweight process and can perform data plane authentication with nominal overhead in a critical scenario. Table 1 draws a summary of recent works illustrated in this section.

4.2. Data plane vulnerability

The data plane devices or switches in SDN are dump devices, and its responsibility is to move the packets based on control plane decisions. The OpenFlow switch properties are (1) one or more flow tables, which is/are responsible for packet movement, (2) input–output buffer for each port, which is responsible for piling the packets before processing. If any flow rule is by now cached in the flow table, then the OpenFlow switch is capable to process the incoming packets directly. Otherwise, a Packet_In message is processed by the OpenFlow switch to the control plane for making the decision. The control plane reply with Packet_Out message for which OpenFlow switch installs the flow rules in the flow table to process (forward, drop, enqueue or modify field) further. Thus, the communication process imposes some delay in decision making that leaves the data plane vulnerable.

The OpenFlow flow table and input–output buffer are limited in size, and OpenFlow switch specifications do not specify any standard for TLS. Moreover, an OpenFlow switch is a dump device and does only packet movement without authenticating the packet. No such authentication or authorization mechanism exists on the switch side. Thus, there is the possibility of a series of table-miss in the flow table. Due to that, the input buffer can receive untrusted data [58]. As buffer size is limited and scaling buffer size is not possible, the input buffer remains full until the control plane decides a packet might cause buffer saturation. Realizing that A. Volkan Atli et al. introduce a buffer or cache management mechanism for incoming traffic over the RYU control plane supports the buffer id mechanism [59]. Before sending the Packet_In message to the control plane, the switch verifies the buffer table for the corresponding incoming packet; if only Packet_In message with unique buffer id appears, it only sends the Packet_In message to the control plane. In another approach, Wang et al. [60] bring architectural modification by introducing cache management between data and control planes. The outcome shows significant flow reduction and efficient flow management in both cases, reducing buffer saturation chances.

OpenFlow switch generates and processes Packet_In message to the control plane concerning new network traffic that appeared for the first time in the OpenFlow switch. By seeing a Packet_In message, the control plane installs the flow rules (drop, forward, enqueue) into the flow table. There is no authentication or validation mechanism in this communication. The consequence of this procedure is that if erroneous traffic flow occurs in the network, the control plane fails to recognize it and erroneously alters the flow table. Moreover, the implementation of Listener Mode in OpenFlow switch might accept unauthenticated connections for configuring TCP Port and also responsible for adversarial rule alteration in the flow table. S. Gao et al. introduce a common platform for both control and data planes to deal with such scenarios. To reduce the control plane's workload, it enforces port control in the data plane and only erroneous traffic flow filtering in the control plane [61]. In case of any table miss, the data plane does not send any messages to the controller despite sending packets to the traffic agent to handle table miss. Such a mechanism of giving some brainpower to the data plane reduces the workload of the control plane. This platform can save more than 80% bandwidth during DoS attacks and can avoid illegal topology updates.

In contrast to erroneous flow alteration, P. Zhang et al. introduce new security primitive named Rule Enforcement Verification (REV) based compressive MAC that allows the control plane to verify whether switches have imposed the rules installed by it, using message authentication code (MAC) [62]. For flow rule enforcement, the data plane introduces the packet-level REV. Similarly, the data-control plane channel enforces flow level REV. After completing the verification flow level, REV compresses all incoming packets of the same into one flow-packet and sends it to the control plane. This compression of flow significantly reduces data-control plane traffic.

Each flow rule has its life span in the flow table and significantly impacts flow table growth. The important issue is that a flow table consists of Ternary Content Addressable Memory (TCAM), which is expensive and power consumer by nature. Due to such features, a flow table usually has a limited size and also non-scalable. These limitations might lead to flow table overflow on the scenario where flow request frequently changes quickly [63]. In a critical scenario, the period of flow rules can appear as a two edges sword for the network performance. The short period flow rule may throw out, which might need for incoming packets. As a result, it will increase table miss events in the data plane and increase the control plane's workload because it needs to maintain each Packet_In message generated by each table miss event. The possible solution is to make a long period for flow rules. However, it has consequences as long periods of flow rules may cause long waiting for other flow rules or send the flow rules in queue or packet loss due to time out. In this scenario [64,65] propose a QoS-aware mitigation strategy.

The research of Yuan et al. [64] suggests the distribution of unused flow table space among the idle switches in the network where each OpenFlow switch is considered a peer in single control plane architecture. Whereas [65] suggest multi-control plane architecture with star topology as in underneath topology instead of peer-to-peer connection, and conveying switch statistics to the control plane implements Discrete-Time Finite-State Markov Chain model (DTMC). The DTMC statistics control plane transmits the flows from overload switch to idle switch and significantly reduces said problem.

The data plane also suffers from flow rule anomalies or rule conflict in the flow table. The SDN control plane applications are developed by different vendors or developers and installed in the control plane to solve their objectives. Each application has its own set of rules to solve its purpose, but there might be a circumstance where two different applications having the same or some extend the same flow rule in the flow table; causes a conflict in the flow table. An intruder can bypass the data plane's security rule and generate some malicious activities due to such flow conflicts. There are many reasons for which rule conflict occurs, such as (1) two or more rules are redundant, which means two or more rules are the same matching and actions. This redundancy might not harm the network, but the elimination of such flow rule is required to make flow table space free for other rules initiation. (2) There may be chances that two flow rules are correlated with each other, where a flow rule might contain some matching with another and vice versa with different filtering actions. Such correlated flow rules create confusion during packet processing. (3) There may be chances for two flow rules to have the same matching but having incompatible flow actions. Such a scenario also creates confusion during packet processing and wastes the flow table resource. (4) There may be a scenario wherein all the matching has incompatible actions, but some network packets match that rule. This scenario invalidates some rules implemented by the network manager/administrator and wastes the flow table resource. These conflicts have a severe impact on SDN architecture as at any point of time intruder can take the advantages of these conflicts to bypass the network security rules and causes devastating consequences in the network. Moreover, the OpenFlow switch specification specifies more than 35 match fields, and each of them having more than 5-tuple, which makes job of network manager/administrator much harder in implementing the rule conflict mechanism, and thus special care is needed for this said problem. In this respect, many research contributions have been proposed among those intended to resolve rule conflict that confuses packet processing are [66–68] or secure rule enforcement mechanism [66,69]. The research work [66] introduces a policy checking mechanism using header space analysis named NetPlumber. The goal is to introduce an agent that sits between the data plane and the control plane. The agent keeps track of every state update utilizing the plumbing graph, which describes dependencies one hop to next-hop and informs the administrator to observe any mismatch. Whereas the research work [67] sits in the control channel and implements an incremental searching algorithm to rectify network invariant violation. The research work by Hu et al. [69] examines flow path space for identifying firewall policy violations. It also uses a header space analysis mechanism for identifying the violations. The authors [70], address the issue of policies and rules anomalies due to control plane modules with miscellaneous objectives. To resolve this, they introduce Anomaly Detecting and Resolving mechanism for rule-level anomalies. To detect anomalies, they propose an interval tree based on state transitions where each node represents a match field, and the leaf node represents corresponding actions. After observing anomalies, they pursue the concept of resetting or remove the rules. The Zhou et al. [71] introduce Customize Consistency Generator (CCG) for customizing network properties using an interface given by the generator. This customization reduces the consistency problem during network updates. Whereas [66,67] fail to do the same. The authors in [68] extend existing SDN flow management and propose a general network model for network state abstractions where each application gets an individual network view. In rule conflict, the model provides a new network state by incorporating the original network states and conflicting rules. Thus, the application does not bother about resolving rule conflict, but it bothers to install new flow rules correctly for that new state.

However, the problem with these approaches is that in any real-time scenario, these approaches fail to enforce violation prevention mechanisms for the complicated rule conflict and creates an open door for security rule violations that can be exploited by the intruder at any period during network communication. Considering such limitations, authors in [72,73] propose two different mechanisms Convert Channel Defender [72] and Transaction-based flow rule Conflict Detection and Resolution [73]) that can resolve complicated correlated or combined flow conflict in the data plane in real-time. For that, Li et al. [72] suggest the implementation of filter rules and introduces a component Covert Channel Defender (CCD) in between the control plane and data plane for tracking the rule modification and prevent rule conflict under Floodlight control plane and Stanford topology-based networking structure. CCD is an extended concept of Veriflow. Both Veriflow and CCD implement a multi-dimensional prefix tree data structure for verification of rule conflict.

Table 2
Summary of data plane vulnerability in existing research works.

Vulnerability	Proposed solution	Implementation	Mechanism	Future work
1. Switch input–output buffer or switch memory is limited in size and rigid in nature. 2. Incapable of making any decision. 3. No TLS standardization is available. Implementation of Listener mood accepts untrusted data. 4. Fake link may be accepted. 5. There is no or weak authentication of data plane and control plane communication. 6. There is no standardized authentication mechanism in the switch for input traffic or incoming buffer data. Thus, erroneous flow alternation is possible. 7. Flowtable memory is also rigid, and no further scaling is possible. 8. Possibility of table miss event increases with a massive amount of incoming data or untrustworthy data. 9. The flow rule time stamp impacts the flow table's growth or table miss event occurrence. 10. Due to a non-vendor specific application running on top of the control plane might create rule conflict in the table. 11. There is no access control mechanism.	1. Proposes buffer or cache management mechanism for incoming traffic [59,60].	1. It implements a buffering mechanism in OvS [59]. Cache management in between data and control plane [60].	1. Switch table miss management of incoming traffic using buffer_id of OpenFlow protocol [59]. The cache uses rate limit, and RR algorithm for incoming traffic [60]. These mechanisms overcome vulnerability 1 and 2.	1. Only support for Buffer_id mechanism, it requires considering of other features [59]. It requires overhead optimization [60].
	2. Proposes port scanning and traffic flow filtering [61].	2. Modifies the data plane.	2. Imposes port scanning to filter out illegal network traffic. It overcomes vulnerability 3 and 4.	2. Optimization of bandwidth required.
	3. Flow rule enforcement verification (REV) method [62].	3. It modifies the data plane and data-control plane channel.	3. The flow level REV uses tag compression using AES and SHA1. It overcomes vulnerability 5 and 6.	3. REV signature implementation.
	4. Unused flow table space among the switches in the network [64,65].	4. Implemented in control plane [64], control plane and Data plane modification [65].	4. QoS aware peer support strategy where a peer is the form of OpenFlow switch. Utilizes queuing theory for switch statistics [64]. Star topology replaces peer-to-peer connection and conveys the switch statistics to the DTMC model [65]. These mechanisms overcome vulnerability 7 and 8.	4. G/G/K model implementation and resistance improvement [64]. Cryptographic verification and large domain implementation [65].
	5. Network state abstraction [68].	5. It extends the Ryu Control plane.	5. Provides abstract individual network view to the application and manages network without interfering with the applications. It overcomes vulnerability 8 and 9.	5. Covers conflict only application flow rules; more, it requires granular conflict addressing.
	6. Identification of rule correlation using multiple packet header [72].	6. Introduce component Covert Channel Defender (CCD) in between control-data plane.	6. Keeps track of correlated rule insertion and modification by multiple message headers and resolve rule conflict before installation dynamically. It overcomes vulnerability 10.	6. Optimization in latency and overhead is required.
	7. Suggests transaction-based authentication [73].	7. Both the control plane and data plane incorporates the modification.	7. The TCDR isolates networks, reduce the conflict and maintain the legality of flows by the authentication mechanism. It overcomes vulnerability 11.	7. Control plane independence addressing required.
Risks in data plane	Confidentiality, Authenticity, Integrity, Consistency, Availability			

Whereas [73] suggest transaction-based authentication where a trust management module running on control plane provides the administrator an interface for which transaction ID and transaction priority can be provided in the network traffic and generate key pairs based on those properties. Another authentication module verifies each flow and stores in the new transaction table using these keys pairs. If any conflict occurs during a network transaction, then assigned transaction priority plays an active role in resolving the dispute. The outcome of these approaches shows optimality in resolving critical conflict in real-time and maintains the flow table's atomicity. Table 2 draws a summary of recent works in this section.

4.3. End-host/control channel vulnerabilities

An end-host in the SDN architecture is a crucial component like a control plane and data plane that constitutes a data plane environment. End-host vulnerabilities need to be determined for the smooth functioning of the architecture. An end-host is nothing but a personal computer or virtual machine or a smart phone or any network embedded computing device [74,75]. Thus, there are possibilities to have several vulnerable applications, like FTP, SSHD, RSH, Nmap are running on the end-host [76]. In SDN communication, the end-host generates the request to the OpenFlow switch for achieving particular resources in the network in the first place. Then, an OpenFlow switch comes into the act and verifies that the requested traffic is already cached or not. If already cached, it directly manages the traffic flow, and if not cached, then a Packet_In message is processed to the control plane for proving the action rule for the respective flow. Here the starting place causes the exploitation of network components as the vulnerable applications are running in the end-host. More importantly, to compromise upper layer components like switches or control planes, an intruder may be much interested in a reachable end-host that is vulnerable and can easily be compromised or malfunctioned (virtual machines). Such compromised end-hosts can overflow the flow table or forced the flow table to advertise fake topologies that turn into the control plane in an unpredictable state [77,78]. Topological adversaries may mislead control plane services or lead to information disclosure of network status. To restrict such adversaries in real-time, S. Hong introduces TopoGuard, a security extension to OpenFlow control plane [79] that effectively detects topological poisoning that mislead network activity.

To achieve the goal, TopoGuard runs an update checker module that dynamically verifies topology updates based on information provided by the other two modules Port Manager and Host Tracker. The Port Manager keeps track of OpenFlow's message and tracks the dynamics of ports for measuring the trustworthiness of a topology update. The Host tracker tracks a host's liveliness in a specific location and provides a forensic judge of host migration. Thus, the collective information of these helps to update the checker to distinguish any adverse activities. In this respect, SPHINX [80] is another generic anomaly detector that identifies inconsistencies in the network states. This approach relies on flow graphs to detect inconsistent data plane behavior and thus fails to scale with a dynamic growth network. Whereas, TopoGuard relies on behavioral profiling and invariant checking for detecting the attack in the network architecture and shows better performance. N. Kaur et al. work on more complicated topological adverse activity where both end-host and OpenFlow switches are compromised [81]. If incoming traffic contains (Link Layer Discovery Protocol) LLDP packets, then before adding a link in the data store, it checks that incoming source DPID and port number pair are in the database or not. If exists then, it verifies that the destination pairs are also the same in both the data store and LLDP packet or not? If not, consider it as a fake link and stop processing.

They successfully resolve this problem, where TopoGuard fails to do as TopoGuard deals with the only compromised host. Therefore, Kaur et al. [81] provide better security in adversarial attacks but fail to detect a compromised switch if it forwards the LLDPs of targeted network switches rather than sending its LLDP to others.

Moreover, the control plane suffers from an unavailable mechanism to ensure the authenticity and integrity of the LLDP packets. Therefore, in [82,83], the authors suggest the implementation of cryptographic unique message identifiers for each LLDP packets and control plane implements hash-based Message Authentication Code (MAC) for each LLPD packet for incoming traffic. The mechanisms prevent LLDP anomalies in the network with nominally increased overhead in the CPU but reduce computational cost compared to topological discovery mechanism.

In SDN architecture, there are no official specification standards for topology discovery mechanism. Therefore, all popularly used SDN control planes implement the de-facto specification standards popularly known as Open Flow Discovery Protocol (OFDP) [84]. The purpose of the OFDP is to gather information about active intra switch links. Since SDN switches are dump devices, there is no intelligence in control mechanisms and automation. Therefore switches are not capable of the link discovery process. The control plane initiates the link discovery process for which each port of each switch gets a dedicated LLDP packet. The major problem with this discovery process is that there is no hard and fast rule enforcement in OFDP that the received LLDP packets should come from switch port.

There are many chances where LLDP packets come from the end-host, which are accepted and forwarded to the control plane. Realizing the fact, Pakzad et al. [85] modify LLDP packets and propose OFDPv2. The proposition includes Port IV Type Length Value (TLV) set to zero, for which it restricts the control plane to send only Packet_Out message for each switch. Upon connection establishment, each OpenFlow switch informs about available ports, port ID, and corresponding MAC addresses. Therefore, one to one mapping can be established between the control plane and the OpenFlow switch. Lastly, PFDPv2 provides the ability to rewrite the packet headers to the OpenFlow switch. This facility can update the Time to Live (TTL) field or Network Address Translation or source MAC of outgoing Ethernet frames. Doing such modification subsequently reduces the control plane CPU load.

The topological adversaries or topological poisoning mainly indicate two types of anomalies: host-location hijacking and link fabrication in an active network. The research works [79–84] are concentrated on mitigating these two anomalies. The authors [86], in their recent research work, evaluated two new threats named Port Amnesia and Port Probing in SDN end-hosts that can bypass the most popular mitigation techniques mentioned above. Port amnesia is a mechanism by which an intruder can achieve the capability of resetting the ports that are used for defending the anomalous link advertisements. Port amnesia opens the door for link fabrication, even if the protection mechanism is there. In port probing, an intruder inquired about a victim host using various active probes from time to time. If the victim host goes offline at any point of time, then the probing indicates this to the intruder, and the intruder starts triggering the host-location hijacking anomalies in the network. Thus, the authors in [86] extend the TopoGuard employing two modules named as a Control Message Monitor (CMM) and Link Latency Inspector (LLI). The CMM is responsible for detecting abnormal control plane interaction, and LLI detects latencies during propagating LLDP packets. The CMM retroactively forces checking procedure for LLDP packets and generate alerts. The LLI uses encrypted TLV for measuring the propagation delay. The inclusions of these two modules successfully overcome the problem.

The above description addresses only two most popular and prime concerned vulnerabilities in the end-host: (a) unauthorized communication between end-hosts (same or different domain) (b) compromised end-host generating malicious flow to adverse the network. Other vulnerabilities exist in end-host that have an adverse effect, such as (c) an insider malicious end-host tries to access unauthorized network services. To protect such unauthorized activity, C. Neu et al. suggest statistical information-based intrusion detection system (IDS) that is capable of mitigating malicious activity even if insider host uses encryption mechanism during communication [87] (d) malicious end-host get free access of networking resources (free access Wireless LAN). The authors V. Varadharajan [88] proposed a policy-driven security architecture capable of identifying and mitigating all the aspects, as mentioned in (a–d) in this section. This proposition conveniently provides network security in the inter and intra domain environment based on a range of attributes such as services and service location, switches, end-host, and security labels allied with the OpenFlow switches and Control plane in diverse domains. Table 3 summarizes the recent works mentioned in this section.

4.4. Section conclusion

Vulnerabilities are the hidden loopholes in the system. If the intruder somehow recognizes the vulnerable point, then the intruder can make adverse activities and restrict the system to achieve its predefined goal. Thus, this section demonstrates component-wise existing vulnerabilities to date in SDN architecture and precautionary measures and the comparison of corresponding mechanisms drawn in this section. In this section, it is observed that SDN vulnerabilities are evolving day by day, and several standard solutions are failed to cope up with the evolutions. For example, the TopoGuard is one standard mechanism to restrict topological poisoning in the control channel, but recently, port amnesia and port probing can bypass this standard mechanism. In the data plane, existing mechanisms struggle with complicated rule enforcement and real-time violation restriction, where CCD is one mechanism that can deal with it. Also, multi-control plane architecture is suggested for reducing both control plane and data plane architecture, but it might increase the overhead in the architecture. It is also identified that maintaining an access control list, transaction-based access control, or role-based access control mechanism produces a comprehensive reduction in vulnerabilities of SDN architecture. Although the precautionary measures successfully resolve all these said problems, it requires continuous observation. By the time of analysis of those, 28 different future works are observed. It also identifies that SDN components opposite to the dispersion of traditional networking and existing vulnerability in the SDN component negatively impact information security in the SDN network. The next section will demonstrate the information security issues due to SDN components' vulnerabilities and briefing the countermeasures.

5. Information security issues and redeems

The information security in the system means that the system must have concurred five properties efficiently (a) confidentiality of the network (b) authenticity in network communication and network components (c) integrity in the network traffic flows (d) consistency in the network and network communication management (e) availability of network resources and services. Due to existing vulnerability in the SDN entities or components, SDN suffers from providing an eminent networking system, and it needs to take special care about it. The opposite architectural dispersion introduces some additional security issues in the SDN compared to the traditional network. This section identifies the consequences of the vulnerability of the components concerning information security in the SDN.

5.1. Confidentiality

The network's confidentiality signifies the prevention mechanism about network configuration information and current communication status from the compromised or unauthorized network components. There is no standardized security mechanism in SDN architecture or

Table 3
Summary of end-host/control channel vulnerabilities in existing research works.

Vulnerability	Proposed solution	Implementation	Mechanism	Future work
1. Compromised end-host. 2. Possibility of advertising fake topology. 3. There is no mechanism for identity control. 4. Vulnerable application may exist on the end-host. 5. There is no specification for topology discovery.	1. Cryptographic host authentication [79].	1. Control plane service improvement.	1. Implements cryptographic host authentication for generating alerts during an attack. It overcomes vulnerability 1.	1. It does not mention regarding consideration of legitimate hosts in attack generation.
	2. Keeps track of topology information [81].	2. It creates two data stores in the Floodlight control plane.	2. The mechanism verifies both source and destination DPID and Port No. for the fake link. It overcomes vulnerability 2 and 5.	2. It requires source verification in the source abstraction domain.
	3. Cryptographic unique message identification for each LLDP packets [82].	3. Control plane modification.	3. It maintains hash-based cryptographic authentication for each LLPD packet for incoming traffic. It overcomes vulnerability 3 and 4.	3. It requires CPU load optimization.
	4. Devices track keeping [86].	4. Control plane modification.	4. It implements device identifiers to restricts control plane abnormal interaction. It overcomes vulnerability 3 and 4.	4. Topology migration attack mitigation.
	5. Policy-driven security architecture [88].	5. Architectural modification.	5. Design policy languages for secure services and communications. It overcomes vulnerability 2 and 5.	5. Implementation of hierarchical control planes using policy inheritance.
Risk in end-host	Confidentiality, Authenticity, Integrity, Consistency, Availability			

no standard TLS implantation mechanism in the control plane and data plane. Thus, when the sensitive information in SDN pass through plain text, it leaves the open door for sniffing sensitive information at a fingertip.

SDN's core communication mechanism suggests that when a packet arrives in the switch and corresponding matching flow rules are not already cached in the flow table, the switch has to send packet information to the control plane. The control plane then had to decide how to deal with that packet for further processing. Thus, it creates a time gap to process the first incoming packet to other corresponding incoming packets. Not only that, much of the time gap is occurred during communications, but a time delay might also impact the time gap to evacuate flow rules from the table and then install new flow rules. During the new flow rule installation, if the flow table is already full then the control plane starts the eviction process using the shortest time remaining policy. Thus control plane needs extra added time for installing new flow. This communication mechanism might reduce the network's confidentiality as an intruder, or a compromised host can disclose the network information by observing such delays. The research works presented in [89,90] justify the possibility of side-channel or information disclosure attacks. The research work presented by Liu et al. [89] is the most sophisticated and advanced approach for gathering the network information as it is not only capable of identifying the current state of flow rule cache but also includes flow rule priorities and existing overlaps in the flow rules. To achieve this, they introduced a replica of the switch utilizing the Markov model that scrutinized the targeted flow rule. The perception of flow rules in the table depends on the Poisson distribution and previous experience. The outcome of this suggests 85% accuracy in the detection of flow rules.

In a different approach, authors J. Sonchack et al. measure the control plane load to identify the network information [90]. In this case, the intruder can probe the network information by sending small packets at a low rate and count the amount of delay added by the control plane. Both mechanisms [89,90] successfully demonstrate that a tiny observation can disclose SDN confidentiality. Later, in an extension of Sonchack et al. [90], the authors propose a countermeasure for the request that consuming a long period to respond by employing a time-out proxy for which the control plane provides a default configuration response [91]. The time-out proxy can be used at the edge of the physical OpenFlow switch. The time-out intelligently keeps a record of each ongoing packet. It installs default forwarding rules concerning a packet for which actual forwarding rules fail to appear within the threshold time, and after installing the default rules, if actual rules have appeared then, proxy discards those rules considering as duplicate entry. Thus, the intruder fails to disclose the information as each flow appears within a threshold time and explicitly reduces the problem. The authors in [92,93] suggest flow rule obfuscation as another effective policy to reduce the information disclosure problem and increase network confidentiality. To create forwarding obfuscation for an incoming flow, the authors in [92] juggle the flow packets into the switches in the network with repeatedly modifying packet header fields and output port. After n − 1 juggling, it selects nth switch for actual flow rule installation whereas [93] randomize flow rule time outs. Due to obfuscation, an intruder fails to determine the actual place of flow generation and thus increases the network's confidentiality.

Each control plane of SDN provides a global view of the network. A single point and centralized control plane architecture or control plane in a distributed multi-control plane architecture can access the network resources and underneath topologies. Thus, any application running on the control plane gain access to the network resources. The consequence is that if an intruder somehow gets access to the control plane or any application running on the control plane might get complete access to network control and do the potential damages to the network. Moreover, there is no authentication mechanism for identifying the participating end-host during network communication at the beginning of the SDN era. By exploiting this loophole, a compromised end-host can mislead network communication or resources.

SDN, at the initial stage, suffered from confidentiality due to vulnerable components. To such incompetency, Wei Wu et al. represent a fine-grained attribute-based signature scheme AC-PORT: an extension of SDN control plane along with two other modules: trusted authority and app-verifier at the northbound interface for authenticating the validity of network applications [94]. In this scheme, each application registers its operation into a signature scheme. Depending on the values registered in the signature scheme, the app verifier identifies the authenticated application. The purpose of this proposition is to protect the control plane from various malicious network operation attacks generated by network applications without a granted access structure. Similar authentication-based schemes are introduced by SE-Floodlight [51], FortNoX [52] and OperationCheckpoint [95]. The Scott-Hayward et al. [95] is the most recent and introduces a restricted set of actions for the control plane application. For that purpose, Scott-Hayward et al. [95] modifies floodlight permission methods, and the permission-checked method is placed in the application-control plane interface and named OperationCheckpoint. Whenever an application generates access permission, then OperationCheckpoint is being called for matching. The OperationCheckpoint identifies the granted permission and restricts or forwarded for further execution.

Whereas, Porras et al. [51] modifies the Floodlight control plane kernel. Similarly, the Porras et al. [52] modifies the NOX control

Table 4
Comparison of confidentiality risk.

SDN confidentiality	Vulnarability	Proposed solution	Implementation	Mechanism	Future work
Control plane	1. There is no standardized security mechanism. 2. There is no standardized authorization mechanism. 3. There is no TLS specification. 4. Global view.	1. Attribute-based access control [94].	1. Extends SDN control plane.	1. The scheme manages the policies to detect the app whose policies against the scheme. It sustains all confidentiality issues.	1. –
		2. A restricted set of actions for the applications [95].	2. Introduce modified actions in the Floodlight.	2. The set of permission determines the application access domain. It sustains all confidentiality issues.	2. (a) Detection of policy violations, (b) Dynamically modification of the set.
Data plane	1. It is a dump device. 2. The packet processing of the first packet compared to the subsequent packets are different. 3. The control plane and end-host communicate with the plain text message.	1. Use a threshold timer for installing flow rules [91].	1. It implements a threshold at the edge of the physical OF switch.	1. The control plane must respond within the time set by proxy or install default rules. It sustains confidentiality issues 1 and 2.	1. It requires to implement in WAN.
		2. Forwarding rule obfuscation [92,93].	2. Make use of programmability features of SDN.	2. Creates obfuscation for inbound network traffic by making a logical change in the flow rule installation. It sustains all confidentiality issues.	2. (a) Automatized obfuscation [92], (b) UDP packets consideration [93].
End-host	1. There is no authenticating mechanism 2. Open communication with the end-host. 3. There is poor TLS implementation.	1. Credential based Authenticating end-host [96].	1. The mechanism implements it as a control plane authenticator application.	1. Provides access control maintained according to the privilege level assigned to the host. It sustain confidentiality issues 1 and 2.	1. Implementation of EAP-TLS.
		2. Hash-based authentication [97].	2. Implement in the control plane over the optical network.	2. Control plane Merkle signature scheme to authenticate end-host. It sustains confidentiality issues 3.	2. (a) Implementation in open-source control planes, (b) Comparison with other authentication schemes.
Threat caused	Control plane hijacking, Malicious applications, Man in middle attack, Eavesdropping attack, Cross path attack				

plane kernel employing a hierarchical authorization role for the network traffic. These modifications also provide digital signature-based validation for authorized network traffic. The difference among these approaches is AC-PORT based on an attribute-based authentication mechanism. SE-Floodlight, FortNox is based on a role-based authentication mechanism, and the OperationCheckpoint depends on the flow rule permissions of each application running on the control plane. The granularity of network operation authentication in SE-Floodlight concentrated on '3' types of roles only, OperationCheckpoint derives '15' types of permissions and the AC-PORT deals with '16' different policies. This observation indicates that AC-PORT provides more granular authentication than the other two to sustain confidentiality.

AuthFlow [96] is another access control mechanism relying on authenticating the end-host. It is a control plane authenticator application. It implements Extensible Authentication Protocol (EAP), where each end-host begins EAP authentication concerning 802.1x standard in contrast to the RADIUS server. It verifies the host credentials, informs the control plane to provide access control, and maps the flow set that belongs to each host. Cho and Szyrkowiec [97] illustrate similar host-based authentication, where the control plane implements Merkle signature scheme similar to x.509 certification to authenticate end-host. During the authentication, the control plane sends a signature and a One-Time Signature key (OTS). Then the end-host verifies the signature based on certification and control plane public key. After verification, the end-host creates its signature using the OTS key and sends it to the control plane. The control plane finally verifies the end-host using the host's public key and the certificate that end-host send. The Cho and Szyrkowiec [97] is the most recent and convincing mechanism to retain confidentiality but is limited to only the optical network. For other networking systems, re-examination is needed. Table 4 draws a comparison of recent works of confidentiality of each plane discussed in this section.

5.2. Authenticity

Authenticity signifies the originality of claimants. SDN provides a global view and opens programmability features within the control plane. Thus, there is no authenticity in the top level of SDN architecture. Any application running on the control plane can have the privilege to access any resources in the network. With the lack of standardized specification of the control plane's network security, the control plane's application can do suspicious activity in the network. Various security-based control plane extension has introduced by the researchers such as ROSEMARY [98], LegoSDN [99] to reduce such incompetency. ROSEMARY improves the resiliency of the SDN by abstracting all the network applications in a close domain and authenticating all the applications simultaneously. The system management module invokes an authentication mechanism during a system call made by an application. This module checks the identity as well as granted permission to the application during a system call. Based on the outcome of this module, the acceptance or rejection is imposed to the system call. LegoSDN improvised by ROSEMARY and introduces the concept of isolating the network applications and networking system rather than abstracting it in a close domain. This mechanism overcomes fault tolerance triggered by events, whereas ROSEMARY fails to address this issue. The sandboxing of network applications generates an RPC call between each application and control plane, and cross-layer transaction provides all semantics or none for the actions that correspond to the event generated by an application. Thus, it improves the resiliency of the control plane, and outcast other [98] by maintaining the network's availability and security.

Nevertheless, the problem with these approaches is performance degradation within a large scale domain. To reduce such limitations, the authors in [100] implement application authentication over Floodlight architecture by introducing an authentication system comprised of a set of access permission and management permission. It offers a log history of illegal access so that unauthorized operations can be identified in real-time. Also, it provides application parameter assessment verifying the dynamic needs of business requests. Whereas A. L. Aliyu et al. introduce a trust management framework [101] comprising of five modules (a) Authentication Module (b) Authorization Module (c) Trust Database (d) Policy Database (e) Monitoring and Evaluation Module. The authentication module comes into the act when any application tries to access network resources. It checks application credentials and consul with the authorization module about the access permission. All the access permissions stored in the trust database and the policy database govern implemented policies. The monitoring and evaluation module manages and reviews trust relationships among applications and control plane every time for anomaly detection.

There is no trust communication relationship between the control plane and the data plane. By default, all the communication credential is moving via plane text in the network. Thus, anyone can sniff the information from the network communication, and damaging the authenticity pretends themselves as legitimate users of this network. Moreover, SDN does not provide any standardized mechanism to identify an authentic host or a malicious host. Thus, any compromised host can pretend to be legitimate and can access control plane functionality or do such unfair activities in the network.

To manage such incompetency, J. Y. Cho et al. introduce a mutual authentication approach of SDN over fiber optics network based on cryptographic hash functions [97]. It also implements a Merkle hash tree where the mechanism comprises of the hash-based signature and tree-based/attribute-based access control. This approach is implemented on networking devices as it does not require access to any complicated cryptographic library such as Secure Sockets Layer (SSL), and hash-based techniques need less computational expense than mathematical operations. An added advantage of this approach is that the hash-based signature can resist quantum attacks so, it might do the same in the SDN environment. Authors K. Benzekki et al. proposed an 802.1x port-based authentication method in the data plane. Introducing IEEE 802.1x facility in a data plane makes SDN architecture more robust, scalable, and secure with less overhead of authentication and verification procedure. It is capable of dynamic assignment of VLANs and increases mobility, enhanced security, and performance by segmentation of the network into multiple broadcast domains [102]. In [103] authors, H. Cheng et al. presented Open Security-enhanced Compatible OpenFlow (OSCO) for OpenFlow switch port security enhancement. This framework mainly consists of a hardware interface, OpenFlow module, cipher algorithm library, and protocol stack. All the module is implemented above the Linux based kernel and constitute secure communication between networking components. It uses a cryptographic algorithm for port encryption capable of imposing security in the network with low overhead. The authors A. Alshahrani et al. also represent their work using a cryptographic mechanism [104]. OpenFlow Switch act as Network Access Enforcer and Access Verifier Service (AVS) implements Ciphertext Policy Attribute-based encryption (CP-ABE) by control plane public key. When an access request from the end-user arrives then AVS imposes an encrypted network policy to the switch. The switch should have a proper secret key to decrypt the network policy for the end-host. Thus, the decryption of network policy verifies the end-host and its platform. Their idea is to use CP-ABE to restrict and authenticate the user for accessing resources in the network. This works well in maintaining latency and CPU utilization compared with previously mentioned techniques. The authors F. Nife et al. introduce a security system that is easily adaptable by the networking architecture and works for both the control plane and data plane [105]. It plots a subsystem authenticator between the control plane and the data plane. The authenticator communicates with the RADIUS server that consists of predefined user credentials and validates an end-user by matching existing credentials. After successful validation, the authenticator communicates with the control plane through an encrypted channel implementing the SSL 3.0 standard. To protect data transmission, it distributes a firewall system in the network for which it is capable of defending the multi-control plane. It can secure the end-host's internal and external malicious activities in the network compared to others.

As SDN architecture suffers from non-standardization of secure communications, so many researchers rely on the implementation of TLS in the protocol architecture. Although TLS is a well-known protocol for secure communications in the network, current research suggests that it also suffers from many attacks such as Transcript Collision Attack [106]. Considering this aspect, B. Agborubere et al. in their research work enhance the TLS in the first place and then enforce this mechanism with OpenFlow to secure the communication between control and data plane [107]. The TLS improvement represents how before the client finishes the connection, it must be reverified by the server using a randomized number and client status verification request within timestamp. The client should reply with an initial hello message ID within timestamp.

The intruder fails to respond within the given time due to the randomization of the number. Thus restrict unauthenticated communication. This mechanism can reduce SDN communication vulnerability and defend from undesirable networking threats like Drown attacks in the future. For secure transmission in SDN, T. Mahboob et al. proposed an authenticating application that runs on the control plane and works for the data plane and other management layers in SDN architecture [108]. This application consists of a hash table, cryptographic hash function, and REST API for access points. The hash table stores the credentials of applications or end-host. Beforehand any communication starts or tries to access networking resources, existing REST API accesses the participating entities' information and validates its access permission. The application is capable of responding instantaneously. Therefore, the administrator/network manager can deal with the malicious device or application dynamically and appropriately. Thus, integrity remains constant with this approach. Integrity maintenance is another challenging issue for information security concerns. Table 5 compares the recent works of the authenticity of each plane discussed in this section.

5.3. Integrity

Network integrity means any networking information remains accurate, steady, and unmodified during its lifetime. The network integrity ensures that no one can change or modify existing rules or services that a network promises. SDN gets a global view of the system by default and the open-programmable environment in the control plane so that anyone can write own code or own functionality in the control plane, and such codes may be vulnerable. Another possibility is that an application running on top of the control plane might be vulnerable. Both the cases open the door for malicious activity or compromise the control plane. More importantly, there is no end-host authenticity in the SDN architecture in its initial stage of development. An end-host directly connects to the OpenFlow switch and is responsible for flow installation in it. A compromised switch can modify the flow rules as there is no such mechanism to authenticate them. It can also do malicious activity in the network by injecting erroneous flow requests in the network. By fundamental SDN rule, control plane has to respond to every flow request. Thus, a control plane has to react to the mistakenly modified flow rules. Therefore, integrity control is critical in SDN architecture due to existing vulnerability and needs special care.

To maintain the integrity in the SDN network, T. Sasaki et al. isolate the control plane's task by incorporating the virtualization mechanism and implements a recovery mechanism to sustain the regularity of the networking system [110]. The partitioning depends on network tenants such that each tenant can restrict itself within one isolated region and can attack itself only. Therefore, this mechanism sustains the network's integrity as network users or applications cannot see beyond their tenants. Moreover, Sasaki et al. [110] incorporate a rollbacking mechanism so that each network component reverts to the initial state and recover the damages. Similarly, the authors in [111,112] recommended a control plane or network state isolation in their research work, and the outcome of their research work suggests isolation of control plane to improve the performance four times better than non-isolation variations. Therefore, virtualization creates different execution environments, which is logically isolated control plane resources over the physical machine and reduces interference between multiple control plane environments. In packet processing, they introduce a

Table 5
Comparison of authenticity risk.

SDN authenticity	Vulnerability	Proposed solution	Implementation	Mechanism	Future work
Control plane	1. Global view or no access control mechanism. 2. Open programmability. 3. There is no authentication for the end-host. 4. There is no authentication for the control plane application.	1. Read–write authoritative permission to each component of SDN [109].	1. Control plane API modification.	1. It delegates optimal API to an administrator for managing read–write restriction users and components. It sustains authenticity issues 1 and 2.	1. It requires more granular privacy.
		2. Introduces isolation or sandboxing network applications [99].	2. Redesign Floodlight architecture.	2. The fault-tolerant architecture explicitly recovers SDN app failures. It sustains authenticity issue 3.	2. It requires end-host state consistency management.
		3. Application authentication and access management system [100].	3. Floodlight control plane application.	3. Uses log security where application certification is verified based on access permissions. It sustains authenticity issue 4.	3. Merging with the intrusion detection system.
Data plane	1. There are no identity checks for end-host or control plane application. 2. There is no mechanism for reliable communication.	1. Introduce an open framework that provides security-enabled interfaces and act as an OpenFlow switch [103].	1. The framework mainly consists of a hardware interface, OpenFlow module, cipher algorithm library, and protocol stack.	1. The hardware interface implements all such facilities while interacting with networking devices and the control plane. It sustains authenticity issue 1.	1. –
		2. TLS handshaking extension mechanism [107].	2. Suggestions only.	2. Before the client finishes the connection, it must be reverified by the server using a randomized number and status request with the timestamp. It can sustain authenticity issue 2.	2. It requires implementation in data to control plane communication.
End-host	1. There is no implementation mechanism for end-host access control. Every new traffic flow pushes towards the upper layer. 2. No mechanism to identify the access control for the unauthorized user from the unauthorized platform 3. There is no mechanism for authenticating the end-host.	1. Ciphertext policy attribute-based encryption [104].	1. Both the control plane and data plane incorporates this.	1. OpenFlow Switch represents itself as a Network Access Enforcer, and in the data plane, Access Verifier Service (AVS) implements policy decryption for the user. It sustains authenticity issues 1 and 2.	1. It requires improved CPU utilization and latency.
		2. Set the level of privileges to end-host [105].	2. Both the control plane and data plane incorporates this.	2. The level of privileges depends on the end-host based on authentication credentials. It sustains authenticity issue 3.	2. It requires overall performance improvements.
Threat caused	Control plane hijacking, Topology poisoning, DoS, Message injection, Fraudulent rule manipulation, Fraudulent rule insertion, Service jamming, Cross path				

software-based packet processing routine that manages Linux network stacks. At the same time, Dwaraki et al. [112] recommended network state isolation. This idea is conceptual, and the implementation remains as future work. In [113], the authors also mentioned network isolation, role-based authentication, and access permission; the use of encryption could maintain integrity efficiently in the network. The authors L. Jacquin et al. propose hardware identity check and low-level configuration monitoring [114]. In support of the proposition, the authors introduce a network verifier that acts as a monitoring plane and connects each network entity to the control plane. The network verifier does remote attestation of network based on network device_id and determines each network traffic's trustworthiness from each network device. It also manages the dynamic configuration of network devices. Thus, only trusted components and verified mechanism could enforce rules. In [115], the authors propose platform integrity maintenance and execution isolation. The authors incorporate Linux Integrity Measurement Architecture (IMA) and Intel Software Guard Extensions (SGE) in floodlight control plane architecture. Linux IMA verifies remote platform integrity by implementing SHA-256 hash-based identity and file control management. Intel SGE is an isolated instruction set architecture where code and data are stored for the respective application and isolated from underlying OS and BIOS. This prototype model does not support the multi-container environment and store integrity in a hardware root of trust.

Integrity control is an essential issue in SDN, and the mechanism mentioned above efficiently leaps up SDN in said issue. If a network can adequately maintain its integrity, then it can achieve consistency in the networking system. Maintaining consistency in SDN is another research issue. Table 6 draws a comparison of recent works of the integrity of each plane discussed in this section.

5.4. Consistency

The term consistency or network consistency means the measure of correctness in the network or network services. A networking system promises critical services such as precedence on path discovery, restricting compromised traffic flow, enforcing avoidance mechanisms for loop, and black hole problems during network transactions. Moreover, due to the increase in demand for network services, a network continually changes its services to cope with demand. The important thing is that the networking system must obey the correctness during such a critical scenario or networking transition. Thus, guaranteeing networking properties during the transition from one correct configuration to the latest correct configuration is considered as network consistency.

Consistency problems might occur in SDN architecture due to rule conflicts such as replication in the flow rule causes inconsistency in networking transaction as multiple rules racing for the same resources

Table 6
Comparison of integrity risk.

SDN integrity	Vulnarability	Proposed solution	Implementation	Mechanism	Future work
Control plane	1. Global view or no access control mechanism. 2. Open programmability. 3. There is no standard for different vendor applications. 4. There is no authentication during rule modification. 5. It misses mentioning application authentication.	1. Isolate control plane execution and packet forwarding process [111]	1. Control plane OS optimization.	1. Using Linux based OS virtualization isolates the control plane execution and packet processing. Thus restricts interference between control plane operations and sustains all integrity issues.	1. (a) To implement in FOG computing, (b) It requires generic optimization techniques.
		2. Network states abstracting framework [112].	2. Application layer.	2. It introduces an intent layer between the control plane and applications that define an API to make available N states to the applications and restricts all integrity issues.	2. Implementation.
		3. Bootstrapping secure communication and execution isolation between control plane and applications [115].	3. Incorporate floodlight control plane architecture.	3. SDN architecture extended by incorporating and Intel Software Guard Extensions for isolating the network management. It overcomes all integrity issues.	3. (a) Multiple Container support, (b) Hardware integrity management.
Data plane	1. There is no specific implementation mechanism for the integrity check mechanism. 2. There is no such verification mechanism for the switch. 3. There is no per-flow installation mechanism. 4. Open communication with end-host.	1. Isolate the task of control plane and TLS certification for communication [110].	1. It incorporates a virtualization mechanism in POX.	1. Creates isolated instances of control plane and data plane by incorporating virtualization mechanism and uses a rollback mechanism to sustain regularity of network. It also uses a TLS key or certification during communication and thus, overcomes all integrity issues.	1. Inconsistent state management during rollback.
End-host	1. There is no end-host authentication mechanism. 2. There is no such verification mechanism for the switch, end-host, and application.	1. Hardware identity check and low-level configuration monitoring [114].	1. Introduces a network verifier that acts as a monitoring plane and connects each network entity to the control plane.	1. This monitoring plane does remote attestation of the network based on device id and determines corresponding network traffic's trustworthiness. It overcomes all integrity issues.	1. It has performance evaluation and manageability implications.
Threat caused	Control plane hijacking, Topology poisoning, Message injection, Malicious application, Network service adversary, Unauthorized access, Man in middle attack, Fraudulent rule insertion, Fraudulent rule manipulation, Eavesdropping attack, Cross-app attack.				

for execution. For flow rule management consistency SDN suggests per-flowrule assignment, but the implementation of such assignment is time-consuming and creates conflict with dynamic requirements for the flow set up or per incident handling. Moreover, such an assignment is hard to implement for those flows having a small life span, thus creating inconsistency in the network. Nonetheless, SDN standardized Barrier command cannot provide adequate acknowledgment about per-flow installation to the data plane or Barrier replies such that the data plane instructs any inconsistency in flow installation to the control plane. Thus, authors in [116] introduced OFLOPS, a framework for measuring capabilities and bottleneck between control-data plane forwarding engines. The OFLOPS incurs seamless interaction between OpenFlow-enabled devices and enables measures to manage control information between the control-data plane channel. OFLOPS improves OpenFlow protocol implementation by providing a generic API for customizing network controls, but it can only quantify the flow activities and network conditions in terms of consistency.

To provide reliable barrier services, the authors in [117] suggest Rule Update Monitoring (RUM), a transparent layer sits below the control plane that captures all barrier commands passing through and holds until switch finish all corresponding tasks. This mechanism ensures the reliability of barrier messages. Bu et al. [118] introduce RuleScope for inspecting SDN forwarding. This mechanism inspects the probe packet's movement between the data-control plane. By default, in SDN, execution of flow rules is based on priority, i.e., high priority flow gets the first chance, but switch treats high priority to the recently installed flow rules. This scenario also creates inconsistency in the network because if a high priority flow is installed first and then comes a low priority flow to the switch, the switch will treat the low priority flow like a high priority and wrongly forward the packets under low priority flow rules. The centralized network view provides dynamically updated network configuration for any critical scenario like topological change, link failure, load balancing, but the problem is that the updating process is not synchronized. Due to this, the network update command might fail to reach all the switches in the network, for which switch might fail to update its flow table accordingly and creates inconsistency in the network [119]. To solve this problem, the Shukla et al. [119] implement WAYUP and PEACOCK scheduling algorithms for policy updates to the subset of switches roundly for each communication. Each round terminates by sending or acknowledging barrier request or response by the control plane or data plane. Once the control plane receives all barrier responses from the subset of switches, the control plane initiates the next round of policy updates.

The inconsistency might occur due to the trustworthiness and standardization of both network applications and end-hosts. A vendor may provide a network application to which there may be other multiple applications available from the other different vendors that might have same or extend the same flow rules for which flow conflict occurred and causes inconsistency. The open programmability also opens the door for rule modification, which might also cause inconsistency. The end-host authentication is not there in SDN specification; thus, a compromised one can modify the flow rules in the switch flow table and causes inconsistency in the network. The Heegaard et al. [120] provide a qualitative analysis of the SDN in contrast to dependability and identifies unwanted cyclic dependency, which is a significant issue for managing consistency.

Many research works have taken place regarding all these aspects. For control-data plane consistency management, the research works are referred in [120–122]. In [122], the authors K. Lei et al. introduced a framework named 2MVeri that deals with the consistency of control and data plane. This framework implements a Bloom filter for the actual processing path during communication concerning a particular switch id and a two-dimensional vector as a tag pushed into the packet header for measuring consistency. This research work claims 100% accuracy in contrast to maintaining control-data plane consistency. The Zhang et al. [121] modifies the switch hardware and software using a new data structure called a path table, which consists of forwarding paths, whereas [122] introduce a 2 × 2 vector matrix to maintain path information. Whenever a packet tries to leave the network, both the control plane and path table verifies the encapsulated header and tag. To achieve the goal, the Zhang et al. [121] implements a Binary decision diagram and Bloom filter-based tagging. In comparison, Lei et al. [122] implements Bloom filter tagging with a vector matrix. In terms of performance, Lei et al. [122] provide less overhead and high accuracy as only the control plane can act dynamically, but in the Zhang et al. [121], both control and data plane handle the forwarding path statically and dynamic updation is considered as future work.

The research works for consistency management in SDN due to flow update is referred in [123–125]. [123], introduce CURE (Consistent Update with Redundancy Reduction), a prioritization mechanism for the switch, which is capable of maintaining consistency during network with incurring less overhead in TCAM. The prioritization depends on underneath topology, packet arrival rate, and existing flow entries. Each switch maintains a counter in the flow table for detailing the matching packets. The authors implement a One-vs-All multi-class classification algorithm for switch prioritization based on the counter value in a single control plane environment. The implementation in a distributed large-scale domain is considered as future work. Wu et al. [125], K. Wu et al. prevent inconsistency of black holes, loops, and network congestions that occurred due to flow updates in the network. The procedure proposes to segment the flows into a short routing path and generate a global dependency graph for locally updated flows to maintain consistency. Then they generate an adjacent dependency graph for actual updates of the flows and, lastly, identifies the occurrence of deadlocks in the graph. If so, then return back to the global dependency graph and start the same procedure again. Whereas [124], decentralized the updated scheduling by means of separating the responsibilities. In this mechanism, control plane initially identifies the dependency of flow segments, generates a dependency graph with switches. The switch then collects network state information from the neighbor switch and schedules the network updates locally if it matches the control plane's precomputed dependency conditions.

The networking environment's rapid growth makes it complicated, and the limitations of single point centralized control in SDN architecture are exposed. Thus, SDN adoption of the multi-control plane is a current trend, and maintaining consistency is an issue. The multi-control plane or distributed control suffers from (1) control plane to control plane communication [126,127]. The research work [126] provides a logically centralized set of Floodlight control planes where each control plane collaborated through a newly introduced modular interface and act as a single control plane by synchronizing their network states. The modularity provides individuality for personalizing the control plane behavior. Three different synchronizing modes: notification mode, service mode, and full mode, are used to manage network updates and resource management transparency. In contrast, Muqaddas et al. [127] represent a cluster of the self-coordinating ONOS control plane to drive Intra control plane traffic. They introduce the anti-entropy protocol and RAFT protocol to the westbound interface for managing Eventual Consistency and strong consistency in the network to reach the goal. (2) multiple control planes cause conflicts in the flow rules [128,129]. To deal with this issue [128] adopted a directed graph approach to detail each control plane's forwarding rules and distributed the details among other control planes by storing them into the graph database (Neo4j). Then inter-network forwarding loop and indirect flow violation problems are resolved, and the graph database keeps track of said information. They also suggest that the blockchain method, along with the graph database, might be incorporated in the SDN environment in the future for resolving the same. However, Lu et al. [129] to reduce the flow conflict implement multiple branch tree-based mechanisms and suggest incorporating machine learning approach in the future. (3) different requirements of applications running top of the control plane [130,131]. In [130], the authors addressed that running an anti-entropy process at fixed intervals causes overhead in the system performance. Thus, the anti-entropy process should be dynamic based on real-time network state consideration and consistency level. They suggest that SDN applications must have a predefined consistency semantics threshold value, and the threshold anti-entropy process reduces consistency risk. This approach works for external application inconsistency. The internal application inconsistency management can be a future consideration in extended work. In comparison, Sakic and Kellerer [131] supports the adaptive consistency mechanism by incorporating a load balancer in the Floodlight control plane. The load balancer provides independent services and supports resource allocation by estimating previous utilization from a data store. Then control plane updates locally utilized resources to a data store for next resource mapping. (4) inconsistency occurred due to network state updates or modifications in the network states [132,133]. The Panda et al. [132] introduce a Superficial Coordination Layer (SCL) in the architecture for which all control planes obey some specific policies in a distributed environment to achieve consistency. The control planes in SCL must have log files that represent the current network state. Each control plane shares their log events to update the network status, exchange probe messages with the switches with the active link, and flow table information concerning each probe message overcomes network failure and maintains its inconsistency. The Sakic et al. [133] also address adaptive consistency measurement where it restricts the control plane application for accessing resources by assigning a predefined consistency level to each resource accessed by the control plane application. In this research work, they introduce the concept of triggers that can dynamically switch the consistency level of resources by comparing per-state-fragment and predefined threshold. This cost-based approach reduces the synchronization problem of resources and maintains consistency. In this work, the consistency is measured based on correctness metrics, but time response analysis is not considered and left as a future consideration. The corresponding references describe the issues and solutions accordingly.

A consistent network can provide good QoS and significantly impact network availability (resources and services). The network availability is another crucial component of information security and needs special treatment. Table 7 compares recent works of consistency management of each plane in SDN.

5.5. *Availability*

In information security, the term availability or network availability means that the probability of service readiness in compliance with network services commitment or specification. The availability metric is highly dependent on the other metrics, as mentioned in this section, i.e., confidentiality, authenticity, integrity, and consistency, and all have a significant rule in achieving the proper availability of the network resources. Network availability is an important property and one of the network administrator/manager's key concerns to reach consumer satisfaction. The SDN is not an exception. The authors, G. Nencioni et al. have done a quantitative assessment to identify the

Table 7
Comparison for consistency risk.

SDN consistency	Vulnerability	Proposed solution	Implementation	Mechanism	Future work
Control plane	1. No proper flow installation mechanism. 2. Flow rules prioritization only depends on the recent entry in the flow table. 3. Control plane applications are not trustworthy.	1. Introduces consistency monitoring tools [121,122].	1. Modifies the switch hardware and software.	1. Implements path table to check traffic behavior and ensure that traffic must obey the path table information [121]. A compressed tag contains path information, and by exploiting the tag, traffic behavior is verified [122]. It overcomes issue 1.	1. (a) Dynamic management of faulty switch [121], (b) Topological comparison is not addressed [122].
		2. Monitor and record the real-time activities of the network, and their casual dependencies between the control plane and the data plane [134].	2. It uses Floodlight control plane.	2. The Foren-Guard statically preprocess control plane applications and dynamically logs the activities of the data plane. This hybridized information constructs a causal relationship between the control plane and the data plane through event-oriented execution trace and state transition graph. The causal graph helps in backtrack system activities and diagnose the forwarding problem. It also provides a command-line tool for declaring queries about the customized and detailed log info. It overcomes issues 1 and 2.	2. (a) It is limited to detecting the flow anomaly but not addresses malicious applications, (b) Proposition based on Java-based application only, (c) It does not mention the accuracy of statical analysis, (d) The mechanism requires incorporating optimization technique.
		3. Implementation of declarative language while developing control plane application.	3. DSML [135] and Prolog [136] is used in control plane application design.	3. The declarative language-based control application automatically and consistently reconciles vendor-agnostic control decisions to resolve application conflicts. It overcomes issue 3.	3. The formal mathematical characterization is required [135]. Prioritizing mechanism of decision-making process [136].
Data plane	1. Flow updates are not synchronous. 2. There is no mechanism for identifying rule modification. 3. No proper per-flow installation mechanism. 4. There is no mechanism for identifying the rule conflict.	1. The decentralized update scheduling mechanism [124].	1. It uses the control plane and the OpenFlow Switch.	1. It runs a scheduling mechanism into the switch based on initial knowledge of control plane configuration. It overcomes issues 1 and 2.	1. It requires to implement in other switches except for the P4 switch.
		2. Dynamic analysis of available resources for each flow segments [125].	2. It introduces a monitoring plane in between the control plane and the data plane.	2. Dynamically determine the available bandwidth and used bandwidth among each segment and update the dependency graph. It overcomes issues 3 and 4.	2. (a) OpenVswitch implementation. (b) Proposed solutions are flow-based analysis, but packet-level analysis might be incorporated [124,125].
End-host	1. Use of asynchronous messages to configure network states. 2. Human error or software bugs may exist in the end-host, which creates policy conflict.	1. Systematically fetching and evaluating network states of OpenStack from the control plane to end-host [137].	1. Control plane modification.	1. The control plane obtains a network state by parsing. An agent collects virtual configuration and parses out the actual network state. Then both informations are sent to the verification server for identification of inconsistency. It overcomes both issues.	1. –
Threat caused	Message injection, Malicious application, Fraudulent rule manipulation, Fraudulent rule insertion, Network service adversary				

similarities between SDN as a backbone network and traditional IP backbone network in contrast to the availability metric. The outcome suggests that the Operational and Management (O & M) failure and coverage failure lead to SDN's adverse effects compared to the availability metric. The impact of hardware and software failure in SDN availability can be manageable by providing a sufficient number of spare processors in the cluster composing the SDN controller [138].

The SDN network provides open programmability, global view of the control plane, but the important thing is that there is no standardized mechanism for authenticating the control plane application by default. So, the basic architecture of SDN is not capable of logical segmentation of applications, i.e., SDN fails to identify each application separately and set the level of access permission. Any failure or error in one or any application or flow rule conflict in multiple applications might compromise the controller and make unpleasant activities such as control plane unavailability to other resources not only that the SDN fails to authenticate control plane applications but also fails to authenticate end-host for a similar reason. By default, the SDN architecture communicates between the control plane and the data plane employing asynchronous messages. Any fault in the control plane application may lead to packet loss during the communications and makes control out-dated concerning the current network status. That might cause unexpected behavior of the network or resource unavailability during communication. Thus, it signifies that access control is one key issue to maintain network availability. In this respect, many research contributors introduce access control mechanism: (1) access control mechanism [53,54] for the control plane. The Klaedtke et al. [53] introduce a permission-based access mechanism to categorize network entities into two categories, such as Subject: the network user and Object: network resources. All network applications run with the permission of subjects. In this case, three different types of permission: read, modify, and subscription to the subject are mentioned to manage the access control. However, Monaco et al. [54] extend operating system principles like Unix and implement a directory-based access control mechanism concerning SDN resource management. The load balancing, congestion control, and security remain a future consideration in this research contribution. (2) Hierarchical access control mechanism [139,140] for control plane application. Both [139,140] propose implementation of stochastic modeling approach for network state management.

However, the difference is that [139] use stochastic modeling with continuous phase distribution to capture the network availability concerning a particular control plane, whereas [140] implement the Reliability Graph for determining the reachability of end-host and stochastic reward for determining the availability metrics concerning particular control plane. In future, the research works might extend by proposing a mechanism for identifying the minimum network resources that contribute to high availability in the network. (3) access control mechanism for end-host [141,142] has been proposed. The Ferguson et al. [141] introduce the concept of hierarchical flow table design for maintaining policy semantics in a tree architecture. To manage policy configuration, they present two more components: a compiler and a database. They map the policy tree to the network flow table, configure the network's policy, and maintain a database to store network component details. The Krishnamurthy et al. [142] suggest a mechanism for conveying SDN switch and application state partition. To achieve that goal, they implement a bipartite graph concept for managing portioning problems where nodes represent the set of switches and state partitions, and edges represent dependency. If dependency, i.e., edges, are more than one, then partitions are subdivided and update the bipartite graph. This mechanism reduces inter-control plane communication. Nevertheless, N. Paladi et al. suggest that providing only access control to the network is insufficient to maintain its availability. They cannot provide a lower level of abstraction and expose network access functionality to the malicious control plane applications. Thus authors introduce an extension of the northbound interface, namely Northbound Access Control API (NACA) [143] and provide a tool for both developer and operator such that a developer can declare their resource requirement for the application and operator achieves the global view of the resources for the deployment of SDN application. The operator has the right to use the security implications of deploying control plane applications and set the limit of network access to the deployed applications by implementing resource-specific policies. Similarly, M. Azab et al. introduce the Proactive Attack-and-Failure Resilience (PAFR) mechanism for isolating end-hosts from the control plane to enhance the control plane's resiliency against network failure [144]. To achieve this goal, it implements Docker: a Linux container for the sandboxing control plane. Once Docker is running, such a control plane can run independently. Thus, the SDN application can share the same control plane or host separately, by which the proposed solution increases the resiliency of the network. The authors in [143] emphasized northbound API abstraction for high availability.

Similarly, but in a different manner, the authors [145] emphasize placing control plane and maintaining southbound API in the network to achieve five-nines availability. Their work suggests that to maintain high availability in the network, multiple control plane architecture is required and, more importantly, placing the control plane in the network to achieve five-nine availability. Thus, this approach implements a heuristic algorithm for control plane placement, which ranks facilities in contrast to the expected contribution to the maximum possible nodes in the southbound interface and determines the control plane's placement. Thus, this approach heuristically minimizes unnecessary control plane connection with the maximum reliability of resources. The outcome shows that 2 or 3 control planes per end-host can reach more than five-nine availability. P. Vizarreta et al. analyze the impact of control plane failures based on Stochastic Activity Networks (SAN) [146]. The SAN model can be represented in the Markov chain and numerically resolve failures, may be software failure or external failure of OS and reliability growth of software. They made comparisons with the open-source control plane (like OpenDaylight and ONOS), and their analysis also shows that a five-nine availability in a single control plane is not possible. Single control plane based network architecture can provide two-nine availability services, and for reaching five-nine network availability, the network architecture must have two control planes for each end-host. The performance evaluation comparisons with commercial control plane remain as future work in this research work.

Introducing multiple control plane architecture for reducing availability problems might incur additional responsibility of load balancing between control planes through east–west bound interface as failing to do so or ignoring to do so might cause frequent network delays and service interruptions. Even as a worst case scenario, it opens the route for an attacker to drain the network resources. To achieve the load balancing in multiple control plane environment, authors J. Cui et al. introduce SDN multiple controller load-balancing strategy based on response time (SMCLBRT) scheme and distributed to each control plane in the multi-control plane domain [147]. The SMCLBRT scheme is a logical load-balancing strategy via switch migration. To achieve the balanced load, it performs a comparison between response time and control planes load. The growth trend of the control plane's response time signifies the load increase. The scheme uses a suitable response time threshold to identify the overloaded control plane and implements a switch migration algorithm to reduce a load of a control plane in the multi-control plane domain in real-time. This research work needs careful observation regarding load distribution pattern, real-time and adaptive selection of the appropriate response time threshold, and its needs to improve migration cost. In another research work, Duy et al. [148] propose an approach for load balancing and failure recovery in the multi-control plane domain and named the approach Aloba. The Aloba introduces the concept of distribution of roles among the control planes where one control plane poses supreme control and is known as the supreme control plane in the architecture, and the rests are regular control plane does the general-purpose task as SDN. The responsibility of the supreme control plane is to estimate a load of regular control planes using the load balancer module and instruct the distribution of loads among the regular control planes. The supreme control plane interacts with regular control planes via the inter-controller channel. The mechanism is implemented using the Floodlight control plane and Mininet. The mechanism is required to test in other control plane domains in the future.

Although every component, along with its properties, is vulnerable in SDN, and we need to take proper care of those, the SDN control plane's criticality is high because of its involvement in controlling the network components during every single communication transaction in the network. Therefore, SDN control plane security should take primary concern for reducing the exaggeration of network status. The authors H. Jo et al. introduced NOSArmor as a security concern control plane [149]. The NOSArmor mainly covers confidentiality, integrity, and availability aspect of information security by integrating eight different security building blocks (SBBs), i.e., security mechanism that covers other two aspects: authenticity and consistency. The NOSArmor provides role-based authentication for marinating confidentiality. It introduces three different roles: admin, network, and management, with different levels of read–write permission of flow, network status, and statistics. To maintain availability and protect malicious user messages, it introduces an OpenFlow protocol verifier. The protocol verifier is responsible for pairing the control messages for appropriate services. It verifies the incoming packet concerning specifications in contrast to message length or field range. Failing to verify, it drops the packet and informs the network administrator/manager by a malformed message. It also adopt network isolation for maintaining network integrity. The difference between NOSArmor and other popularly known secure SDN control planes like Rosemary, LegoSDN, SE-Floodlight is that other secure control planes deal with specific SDN issues. Like Rosemary manages authenticity, LegeoSDN manages authenticity and availability, SE-Floodlight deals with confidentiality only, but NOSArmor directly or indirectly covers all information security issues.

Moreover, it shows better performance than the others. The only limitation is that it is implemented in a single control plane domain and requires concentrating on large-scale domain issues. NOSArmor is the most prominent approach for information security, according to this study. Table 8 draws a comparison of recent works of availability of each plane discussed in this section.

Table 8
Comparison of availability risk.

SDN availability	Vulnarability	Proposed solution	Implementation	Mechanism	Future work
Control plane	1. No or weak authentication mechanism for control plane application, incoming traffic, and end-user. 2. No or weak authorization for control plane application, incoming traffic, and end-user. 3. There is no specific level of access permission. 4. There is no lower-level abstraction. 5. Control plane resources are rigid. 6. Insertion of malicious codes or loops.	1. Declarative access control of network applications [143].	1. It extends the northbound interface.	1. The northbound extension act as a tool for declaring the required resources for applications during any access request. It overcomes issues 1, 2, and 3.	1. (a) Performance improvement, (b) Overhead reduction, (c) Taxonomy of actions applicability and (d) extension integration in other control planes.
		2. Control plane placement such that each node optimizes a path with minimum cost [145].	2. Architectural Optimization.	2. Implements a heuristic algorithm for control plane placement; thus, minimizes unnecessary control plane connection with maximum reliability of resources. It overcomes issues 3 and 4.	2. It requires addressing the out-of-band network control.
		3. Stochastic Activity Networks model (SAN) for control plane [146].	3. Formalize SDN architecture based on SAN.	3. Formalize SDN architecture based on SAN. The SAN provides facilities of marking the states, and instantaneous activities change upon system state using the numerical method by converting the model into a Markov chain. It overcomes issues 5 and 6.	3. (a) Impact on network services, (b) Implementation on the commercial control plane.
Data plane	1. The flow table is rigid, and no possibility of scaling further. 2. Memory is also rigid in nature and expensive. 3. Flow updates are not synchronized or weak exception handling. 4. There is no mechanism for network traffic identification. 5. There is no upper layer abstraction. 6. There is no mechanism to restrict the end-host.	1. Multi-level flow table architecture [150].	1. TCAM flow table works with coordination SRAM flow table.	1. The incoming packets search for flow entries in the TCAM table in the first place, and if entries are not present, then the search is continued with the SRAM table. If both cases fail, then only packets are sent to the control plane for flow entries. It overcomes issues 1 and 2.	1. Implementation.
		2. Routing aggregation [150]	2. Routing algorithm.	2. Compress the multiple flow entries without changing the next forwarding hops. It overcomes issues 3 and 4.	2. Implementation.
		3. Allow flow requests that follow the complete TCP handshaking [151].	3. Switch architecture modification.	3. Introduce a connection migration module and an actuating module in the data plane. The first module test the flow request completes the TCP handshaking or not and stops processing failing to do so. The second module accelerates network status to the control plane. It overcomes issues 5 and 6.	3. It requires an implementation mechanism other than TCP and also requires to minimize the collateral impact.
end-host	1. Tight coupling between the control plane and underlying topology/end-hots. 2. Lack isolation of underlying end-host from the control plane.	1. Introduce a lightweight sandboxing mechanism by incorporating Linux containers [144].	1. Implements Docker: a Linux container for sandboxing control plane.	1. The Docker provides an environment where users can customize or deploy packages for the control plane and encapsulate them from others. It overcomes both issues.	1. It should incorporate a dynamic mitigation mechanism for fake control plane cloning.
Threat	DoS, Resource saturation, Service chain jamming				

5.6. Section conclusion

This section demonstrates the pinpoints about the information security properties by enlightening background component vulnerabilities. It also includes corresponding existing countermeasures and highlights the mechanism and the modifications introduced in the architecture. In the case of existing countermeasures, it is observed that credential-based mechanisms (attribute-based, role-based, host-based) are the dominant mechanisms to maintain confidentiality and authenticity in the network. The enhancement of TLS suggested by B. Agborubere et al. is another prominent solution, but it requires thorough observation and practical implementation. To maintain network integrity, network property isolation (control plan, network state, or network) is the key mechanism. The suggestion of fusing the role-based authentication with network isolation by Dave and Nagaraju and the suggestion of execution isolation by Girtler and Paladi are the two notable recent

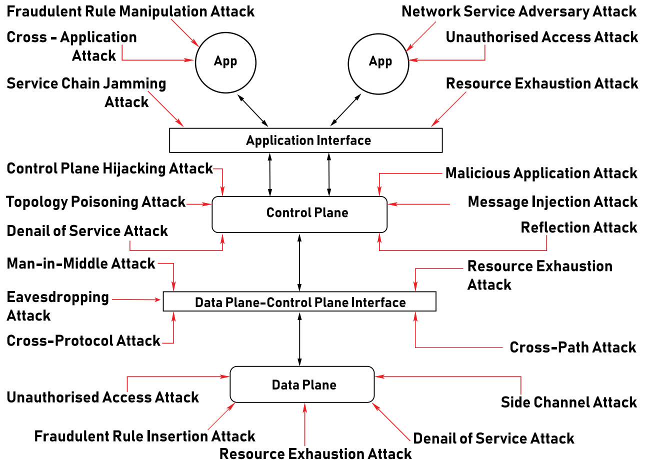

Fraudulent Rule Manipulation Attack
Network Service Adversary Attack
Cross - Application Attack
App
App
Unauthorised Access Attack
Service Chain Jamming Attack
Resource Exhaustion Attack
Application Interface
Control Plane Hijacking Attack
Malicious Application Attack
Topology Poisoning Attack
Control Plane
Message Injection Attack
Denail of Service Attack
Reflection Attack
Man-in-Middle Attack
Resource Exhaustion Attack
Eavesdropping Attack
Data Plane-Control Plane Interface
Cross-Protocol Attack
Cross-Path Attack
Data Plane
Unauthorised Access Attack
Side Channel Attack
Fraudulent Rule Insertion Attack
Denail of Service Attack
Resource Exhaustion Attack

Fig. 3. Layer wise attack possibilities.

works in this case. For consistency management, 2Mveri and CURE are recent works those provide high accuracy but fail to achieve the same in multi-control plane architecture. For a multi-control plane, the architecture-directed graph approach to manage the control plane's forwarding rules and adaptive consistency management mechanisms are prominent. Lastly, NOSArmor is the recent and prominent work to maintain the availability, but the critical thing is that NOSArmor has the versatile capabilities to maintain other properties of information security. The comparison of recent countermeasures identified 49 different future works from this section. While escalating the information security, the ongoing observation identifies that SDN suffers from 22 various practical threats in a live network; among them, cross-app, cross-protocol, cross-path, and reflection attacks are the most recent. There are few contributions available regarding those attacks, and these need special care in the future. Not only that, the ongoing research study also identifies many other issues over SDN architecture. The upcoming section addresses those threats with the incompetency of Information Security properties and possible countermeasures.

6. Information security threats and redeems

Following the above discussion about vulnerability and information security loopholes in SDN architecture, we divide threats into five categories based on the characteristics and corresponding targeted layer, i.e., control plane, application plane or application layer, application to control plane interface, data plane layer, and control plane to data plane communication interface. Fig. 3. depicts the layer-wise separation of all the attack generation, and the detailed attack scenarios are explained in the following Sections 6.1 to 6.7.

6.1. Threats in control plane

The control plane is most vital as well as the most significant entity of SDN. The control plane is considered as the heart of SDN and it is the central point of attraction of the intruders or malicious users. Yu [152] presents a transcript of a general discussion regarding the importance of the SDN control plane. It mainly addresses the importance of authentication of application in the control plane and is highlighted for the trusted functioning of SDN. The threats those SDN control plane faces are discussed below:

6.1.1. Control plane hijacking attack

The control plane hijacking is an attack where an intruder gains access control of the network with or without the end-user interventions. The lacking of confidentiality, authenticity, and integrity makes SDN vulnerable. For example, an application running on top of the SDN control plane gains access control underneath architecture. If a malicious application with the same capability executes system exit instruction, causing not only the termination of the application but also the control instance, in other words, hijack the control plane from the underneath data plane. Therefore, access control for the control planes and the application must be required to prevent such unwanted activities in SDN. The authors Y. Tseng et al. present ControllerSEPA, a lightweight control plane independent plug-in to secure SDN from illegal access of control plane application [153]. It is more like a REST-API system with more functionality compared to REST-API. It keeps the records of OpenFlow applications as well as rule insertion by the app. For rule verifications, it runs a switch-based rules verification (SRV) or Alias Set Reduce (ASR) algorithm. This research work also implements a set of access permissions for which it restricts the scope of resources for the applications. Byzantine fault-tolerant (BFT) mechanism [154] is another solution for the control plane hijacking attack. The BFT provides a robust control plane replication such that network resilience remains intact during failure. The authors in [154] deploy the replicas into the cloud and implement a cost-effective control plane assignment algorithm to select minimum multiple control planes to reduce such a problem. The proposition represents a prototype only and a full-fledged model test in the real world, and identifying appropriate BFT remains future work.

To provide steady performance, SDN should enhance its security measures. In this respect, authors H. Cheng et al. introduce Open Security-enhanced Compatible OpenFlow (OSCO), an open security platform of SDN [155] supported by Raspberry Pi Single Board Computer (SBC) hardware. It extends the OpenFlow protocol by incorporating a cryptographic algorithm and ensure a secure communication mechanism between the control plane and data plane. The research work implements a block cipher and hash-based algorithm, and the outcome depicts that OSCO can reduce undesirable events in SDN. OSCO is a powerful mechanism in this respect, but hardware independence is an issue for rigorous implementation and may be considered in the future.

6.1.2. Topology poisoning attack

The lack of authenticity and integrity control in SDN architecture opens the door to form a topology poisoning attack in this domain. This attack includes forging network links or network packets and unwanted modification of the underneath topology. S. Hong et al. [79] addressed such an attack for the first time and demonstrated that the implementation of Host Tracking Services (HTS) and Link Discovery Services (LDS) in SDN opened the door for poisoning the topology. They proposed TopoGuard for reducing topological poisoning. The TopoGuard is the most popular defense mechanism for topological poisoning attacks. A current research work carried by S. Xiang et al. verify TopoGuard with Process Analysis Tool-kit, a popular model checker, and Communication Sequence Process, a well-known process algebra [156]. The outcome of their analysis shows that TopoGuard fails to detect host hijacking if an attack is generated during the downtime of the target host's migration and leaves an open issue for the researchers. E. Marin et al. undergoes a similar investigation over TopoGuard, and TopoGuard+ [157]. They identify two vulnerable surfaces in these existing mechanisms: (1) insecure mechanism to track link latencies and (2) lack of freshness in LLDP packets over the said mechanisms. Exploiting the pitfalls of these mechanisms, a malicious user can generate two new attacks (i) Reverse Loop: by exploiting the attribute LINK-Type of the Link Discovery Service (LDS), a malicious user can extend the LLDP round trip time as much as possible such that it exhausts the resources of the control plane and produce devastating consequences for the control plane. (ii) Topology Freezing: exploits LDS's topology management module and prevents the control plane from updating the network status and leads system runtime exceptions in the network. To restrict these two attacks, the authors propose (a) the MAC-tag field of LLDP packets comprised of DPID, the port number, and time stamp utilizing a single cryptographic key instead of different key LLDP packets in every round trip. (b) Link Latency Inspect (LLI) module should extend latencies for the genuine links and suggests to incorporate distance boundary protocol to verify the upper bounds between participating entities (adversaries and LLI) (c) to extend HTS by incorporating a middlebox in between control plane and data plane such that control plane verifies the end-host for every packet_In request. It is worth mentioning that all these propositions are perceptual only and leave the open field to carry out future work. The authors consider Floodlight control plane architecture to make the suggestion. Thus, there is a fair chance to do similar research on other control plane architecture in the future.

6.1.3. Denial of service attack

This attack is the most common and famous in the SDN control plane due to its centralized control over the network. Due to weak or no authentication of incoming packets in the network, resource availability increases the possibility of crafting this attack. In certain circumstances, if an excessive number of incoming packets occurred in the control plane and the control plane uses all its available resources to route such enormous packets might lead to an unpredictable or unreachable state of the control plane. In this respect, Kotani and Okabe [158], Ambrosin et al. [159], Han et al. [160] and Wang et al. [161] are the recent and popular mitigation techniques for single or multi-control plane architecture. Kotani and Okabe [158] propose a packet filtering mechanism where each switch can maintain a record of the packet header of each Packet_In sent out and discarded duplicates. The mechanism consists of two sections: (1) Pending flow rules specify the header fields to which the control plane refers and (2) Pending flowtable where Open vSwitch records the header fields. This work does not address the efficient use of the pending flow table and lets an open place for future improvement. The authors in [159] propose a probabilistic proxying and blacklisting mechanism for restricting network traffic to overcome the exhausting control plane resource problem. The mechanism is an Open-Flow module named LineSwitch and resides in the OpenFlow switch's edge for restricting SYN flooding dynamically. Therefore, it imposes less overhead compared to Kotani and Okabe [158] by distributing module management into two sections. The only limitation is that it deals with SYN flooding and needs to address other future attacks. In another approach [160], authors introduce an SDN framework that provides collaborative intelligence to the control plane and data plane. To make the data plane intelligent, it incorporates a monitoring algorithm on the data plane side and an attack classification mechanism on the control plane side. The classification mechanism runs the machine learning approach by incorporating autoencoder and softmax classifier. This approach can reduce SYN and UDP as well as SYN and ICMP flood. Wang et al. [161] introduce the SafeGuard Scheme, where this scheme emphasis a similar concept of Han et al. [160] but with a different mechanism and multi-control plane architecture. The data plane monitoring system identifies malicious traffic by four tuple vectors (byte rate, symmetric flows percentage, the variation rate of asymmetric flows, and flow percentage with small amount packets) back propagation neural network method. The control plane defense implements a master–slave approach for remapping the control plane and send access control messages to switches.

6.1.4. Message injection attack

The weak integrity control, consistency, and authentication mechanism increases the possibility of crafting this attack. This attack severity directly affects the network service that it promises, consumes the maximum amount of network components, and imposed topological adversary. A. Shalimov et al. address such an attack by injecting incorrect message header length in the OpenFlow [162]. Such injection is that it does not harm the control plane but discontinues the switches that send the packets. In a most recent research work PacketChecker [163], propose a lightweight module for a control plane (floodlight in this case) that can detect and mitigate the falsified packets. This approach observes MAC address, switch DPID, and imports information carried by Packet_In message header to distinguish injected packets. After identifying packet injection, it sends a flow-mod message to the switch for discarding the packets. A. Alshra'a et al. introduce hardware extension in SDN architecture, namely the INSPECTOR device [164], that authenticates the incoming packets and reduces the packet injecting threats in SDN. Compared to the previous mechanism [163], the INSPECTOR hardware component is introduced in the edge of OpenFlow switch and store all Packet_In message header as well as comparison mechanism. This mechanism's advantage is that it can isolate the control plane for which control plane performance will not be affected during filtering. However, major incompetency is that the INSPECTOR itself is suspectable of compromised, which leads to future consideration.

6.1.5. Malicious application attack

The Network Operating System of SDN, namely control plane or controller, provides open programmability and global access. The lack of authentication and vendor independence increases the possibility of malicious applications running on top of it. The default SDN architecture also suffers from confidentiality, integrity, and consistency control over the network, and using such deficiencies, a malicious application may constitute many adverse activities in the network. A simple example of such activity is that a malicious application can change the system time and stops many transactions in the network. Thus, the identification of malicious applications is an important criterion for the control plane. Considering this issue secure operating system NOSArmor [149], ONOS [165] are introduced in SDN architecture. The Röpke and Holz [165] is a sandboxing system that can restrict network components and applications to access sensitive configured operations. To implement a sandboxing mechanism, ONOS incorporated OSGi technology, i.e., OSGi conditional permissions, and tested in both HP control plane and OpenDaylight control plane and whereas NOSArmor proposes '8' different SBB blocks to represent secure OS.

In contrast to the performance issue, NOSArmor shows better performance than ONOS as it degrades performance when the number of switches is more than 32 [149]. The authors J. Noh et al. introduced a permission-based detector [166] for identifying the malicious applications and preventing the execution without permission. To achieve the goal authors maintain a permission table for each application, and the behavioral pattern of the user application is clustered based on their functionalities. The mentioned mechanism is implemented in the Floodlight control plane only, and adopting the network's state remains future work.

6.2. Threats in application layer

6.2.1. Fraudulent rule manipulation attack

The control plane applications are a vendor-independent SDN component and have global access to other SDN architecture components. Due to lack of authenticating applications and lower-level abstraction, integrity control and consistency control mechanisms cause such an attack. The notable observation identifies that a malicious application can create rule conflict or overwrite existing flow rules and cause unexpected network behavior. A malicious application can generate enormous control messages that flush out the actual flow entry from the table, which can also mislead the network behavior. The authors C. Röpke et al. introduce a mechanism that can compare the actual network state with the compromised state, and it can also drop adverse network activities before their implementation [167]. To achieve the goal proposed security mechanism consist of four components: an observer, an SDN control plane stub, a comparer, and a message store. The observer intercepts control plane messages, and the control plane stub captures the network view and sends both information to the comparer where the messages are stored and compared with previously sorted information for identifying misleading flow installation. This mechanism can handle compromised control plane and malicious application for flow installation. In another approach, C. Qi et al. propose a multi-control plane environment where multiple control plane is responsible for deciding the proper flow rule installation [168]. They suggest that there should be more than three control planes with similar privilege over underneath subnet in the network. For scheduling the control plane, it introduces four virtual functions: Transponder: for gathering network and network state information, Sensor: monitoring network state and identifying abnormality in the network, Decider: decide control plane benign conditions, Scheduler: selecting master control plane for a network. Since multiple control planes participate in judging the master control plane for a network, the difficulty level increases for flow rule modification compared to a single control plane network, thus reducing the risk of flow rule modification attack.

6.2.2. Network service adversary attack

A malicious application can obstruct many networking services in the network. Lack of consistency and integrity control leads to such obstruction. A malicious application might contain some arbitrary code that can hijack the control plane or carefully drops packets for particular incoming packets from specific switches. A malicious application might intentionally execute infinite loops that make the control plane busy for an indefinite time or block the specific network services for an indefinite time. The consequence of such a scenario in SDN may be the dropping of a legitimate packet, or a flow rule may be flush out as each flow has a specific span of life. Röpke and Holz [169] demonstrates that SDN rootkit easily can bypass the sandboxing mechanism. Not only that, but it can also bypass the policy checker mechanism as described in [66,67]. Therefore, protecting malicious application is an important issue and realizing that the Röpke and Holz [165] propose a sandboxing mechanism that can reduce malicious rule modification. This mechanism unable to verify network level operations verification and limited to system level operations verification. Also, it cannot perform detection and protection simultaneously. Very recently, the authors in [170] introduce a proactive mechanism that statistically analyzes the application behavior and automatically detects the misleading activity by the machine learning approach. They implement Security Sensitive Behavior Graph (SSBG) for extracting the control plane semantic features and behavior profiling. The extracted features use unsupervised machine learning (k-mean) algorithm to detect malicious applications. This approach statistically determines network behavior. It can be made more efficient by incorporating real-time detection in the future. Moreover, it introduces only a detection mechanism, but the suggestion of a mitigation mechanism is also an issue to cover.

6.2.3. Unauthorized access attack

An application has the permission of the global view of the network. Due to a lack of integrity and consistency control, unauthorized access incidents might be possible. A malicious application can do adverse activity in the network taking advantage of such permission. The Porras et al. [51], Scott-Hayward et al. [95], Yu [152] and Wen et al. [171] are research solutions that deal with such application-based attack. The Wen et al. [171] provides '18' different permission for API entry in the control plane. Those '18' permissions are categorized as Read: to manage sensitive information, Notification: to manage real-time events, Write: to manage state modification of control plane or switch, System: to manage access local resources by an application. It also introduces an access control layer to restrict the direct interaction of applications and control plane OS. This mechanism successfully reduces unauthorized access.

6.3. Threats in control plane and application layer channel

6.3.1. Resource exhaustion attack

This thorough study observes that a malicious application causes network service adversary, and due to this, it may consume resources or drop packets to obstruct control messages of other applications in the network. The determination of mitigation is in future work.

6.3.2. Service chain jamming attack

The SDN applications generally execute a sequence of services, which is also known as the service chain of an application. The network service adversary may also block the service chain of an application. Thus, the correctness of network applications is much required to resolve these issues. NICE [172] is an automated tool that verifies the correctness of network applications. It runs a model checker for explicitly state checking. However, the model checker fails to discover the entire state space as it is unpredictable to specify. Thus, NICE incorporates the domain-specific heuristic search for reducing the space of event orderings while discovering invalid states. In the future the work may be extended and a more sophisticated search algorithm might produce more significant outcomes.

6.4. Threats in data plane

An intruder posed an eagle eye on the data plane of SDN architecture as it is the component directly associated with the end-host, and utilizing such components (data plane and end-host), an intruder can generate many attacks to the system. The threats that data plane can pose are:

6.4.1. Unauthorized access attack

An end-host connected to the data plane can lead to such adversaries in the network due to integrity and consistency control in the data plane. The Menezes and Duarte [96] is dealing with this adverse effect based on authenticating the incoming flows in contrast to the credential of the end-host.

6.4.2. Denial of service attack

This attack is the most severe and actively used threats in SDN architecture. The lack of information security features impacts the data plane for causing DoS attack or flooding attack. Dumpiness and small memory of the data plane make it easy to forge the packets using compromised end-hosts and exhaust the memory or flow table. Dealing with a programmable data plane is suggested in [173]. The research work incorporates P4: a data plane programming approach to characterize the entropy of IP address for every incoming packet. The detection unit calculates the dynamic threshold by taking this value as input and generates alarm by exceeding the last entropy estimation, whereas [174] suggest a trigger-based algorithmic approach. This mechanism runs a monitoring algorithm and compares the number of Packet_In messages to a threshold value. In exceeding the threshold value, it triggers the detection function. The detection function consists of two sections: host-defend function: counts IP address to a threshold and switch-defend functions: count the switch to a threshold. This mechanism implements Ryu as a control plane.

In comparison, the Lapolli et al. [173] produce a better result as the threshold is dynamic, not predefined like [174] but incur computational overhead to maintain dynamism. In the more recent research, Myint Oo et al. [175] implement a support vector scheme. The other two [173,174] approaches work on single control plane architecture, but this approach implements a cluster of control planes. On arrival of packets, switch verifies packet concerning flow entries and send Packet_In message to OpenDaylight control plane cluster if any mismatch occurred. Then the control planes run Advanced Support Vector Machine (ASVM) based DoS attack applications for classifying the packets abnormality. The ASVM application recognizes the average flow or abnormal flow for the incoming traffic. The work leaves the scope for incorporating real-time DoS detection in the future.

6.4.3. Side channel attack

In this type of attack, the intruder analyzes the time gap or the flow configuration delay in the flow table to fetch the network configuration. Due to the lack of confidentiality in the data plane, this type of attack is possible in SDN architecture. In this respect, many Fingerprinting approaches [176–178] have been proposed. The authors in [176,178] demonstrate the circumstances and the mechanisms for which an intruder can identify the acting control plane for a networking transaction. The studies in [176,178] demonstrate that analysis of time difference between sending and receiving packets leads to information leakage about the network states to the intruder. The Azzouni et al. [177] extend a similar concept along with the indication that analyzing the LLDP or ARP packets in SDN can also lead to similar incompetency of SDN. The outcome of this research work proves that the time-based features empower an intruder to digs out critical information regarding flow rule configuration and infer the security policy of the SDN. All these references indicate the possibilities of a side-channel attack in SDN, but mitigation techniques lead to open issues for future SDN architecture considerations.

6.4.4. Fraudulent rule insertion attack

As there is no or weak authentication, integrity, and consistency control in the data plane, it makes it easy to lodge this attack. If a malicious application or a compromised host generates fraudulent flow rules, then in the data plane, no such information security mechanism exists that restricts the insertion of such rules. This type of insertion causes a network poisoning attack or unexpected behavioral change in the network.

6.4.5. Resource saturation attack

SDN data plane suffers from availability as resources are limited in size and not scalable. Moreover, there is no or weak authentication mechanism that exists in the data plane. Thus, saturating the data plane is an easy one for the malicious user. In this respect, the authors [179] present an event handler evection model. This model comprises two modules: (1) learning switch module: analyze the switch statistics and add or remove flow rules based on statistics or timeout events. (2) flow checking module for validating the flow rules by verifying logs and block them. Not only that, it removes the flow if it exceeds the packet/sec threshold. The authors in [180] have made a thorough analysis regarding such an attack in the data plane and identified that even though [179] can reduce saturation attack but fails to recognize sophisticated attack in the data plane. Thus, they recognize and propose three traffic features: Growth of Foreign Flow Consumption (GFFC), Deviation of Flow Amount (DFA), and Commonness of Flow Entry (CFE) to differentiate abnormal flows and also determine the attacker. They also include a mitigation mechanism using a token bucket module where a token is added into the bucket or removed from the bucket based on flow entry and exit. The rate of token entry determines the average transmission and able to reduce resource saturation attacks.

6.5. Threats in data plane to control plane channel

6.5.1. Man-in-Middle attack

The Man-in-Middle attack is another popular attack scenario in the networking world where an intruder tries to sniff the data during communication by injecting a host in between source and destination. The data to control plane communication is prone to such attacks as communication uses plane text, no or weak confidentiality, and integrity mechanism exists in the communication channel. M. Brooks et al. justify the possibilities of MITM attack in SDN architecture [181]. The authors demonstrate such attacks over the OpenDaylight control plane and try to generate an attack with the help Ettercap tool of Kali Linux. Using this tool, they successfully capture the credential of the DLUX web interface. The DLUX feature is not a compulsory component but needs special care while implementing as it leaves a weak spot in the architecture. In very recent work, the authors K. Zhang et al. introduce a detection mechanism named CMD to detect MITM attack in SDN [182]. The authors propose a Global Flow Table (GFT) that captures the identification of each flow entry, matching entry field, number of bytes, and packets in each flow, creation, and deletion of time. The GFT, along with network topology and connection establishment in the communication, detect the MITM attack. The real-life implementation is not mentioned and remains in future consideration.

6.5.2. Eavesdropping or sniffing attack

The data to control plane channel is prone to eavesdropping due to confidentiality and integrity mechanisms not appropriately implemented in SDN architecture. Authors in [183,184] address this issue and provide a multipath solution for this attack scenario in SDN. The Aseeri et al. [183] addresses the two-way multipath mechanism. The mechanism runs the Dijkstra algorithm to identify the shortest path between the source and the destination and instruct the sender to send acknowledgment via the same path. It influences the hard timeout and idle timeout in the flow rules to control path switching and consequently disable ACK blocking by an attacker. Whereas [184] address the rule priority in multipath mechanism and influence only idle time restricting eavesdrop in the network. They also encourage implementations of the capacity scaling algorithm in place of the Dijkstra algorithm for increasing the optimality in finding the shortest path.

6.5.3. Resource exhaustion attack

The data plane resources are limited and suffer from the availability problem. Thus, it is relatively easy to exhaust the channel by flooding the incoming packets or using many forge Packet_In messages. In this respect, FloodDefender [185] incorporates an attack detection module for continuously monitoring the network status, table-miss engineering module for offloading some table miss packets if it detects an attack scenario in the first module. It also incorporates a two-phase filter: (1) frequency-based filter: drops high packets with low frequency, (2) traffic-based filter: drops packet using feature classification of incoming packets. The last module implements flow rule management to eliminate most of the flow table's inadequate flow entries. Thus, all these modules collaboratively mitigate communication bandwidth and computational resource exhaustion attacks. The limitation of the work [185] is that all the modules execute from the control plane and deal with both table miss and Packet_In messages. Thus, these impose extra overhead in the control plane. In essence, Zhang et al. [186] propose a better lightweight mechanism named FloodShield, where the source address validation module filters out forged packets by explicitly installing filter flow rules. The second module recognizes the legitimate sources that can also do malicious activity. Thus, this module encounters statistical analysis based on traffic features and dynamically restricts the network resource usages. In practice, the implementation of source address validation in conjunction with SDN and traditional network, i.e., representing a gateway, might be considered in the future work.

6.6. Other attacks

While reviewing the attack scenarios of SDN, we observe that there are few more attacks currently trending in SDN architecture and required much more consideration in the future. We address these attacks in this section and address the current solution we found during this research study.

6.6.1. Cross-Path attack

The Cross-Path Attack disrupts the shared links in between data to control plane channel during communication. A well-crafted small amount of data traffic is pushed into shared links to implicitly hinder the delivery of control messages before reaching the actual data plane messages. Due to lacking authenticity, integrity, and confidentiality, this attack is possible in the SDN architecture. The authors J. Cao et al. addressed this issue in the first place and demonstrated some possible countermeasures: Delivering control messages with high priority, Proactive reverse bandwidth, and Distributing Path Reconnaissance [187]. Those proposed countermeasures indicate to deal with such a novel attack, but the implementation of such propositions remains an open issue in the network for future consideration.

6.6.2. Cross-App attack

This attack takes advantage of the weak or non-existence integrity control mechanism of SDN. In the Cross-App attack, an application poison the network integrity by stealing other applications' authorization and starts actions on behalf of other applications. The current flow rule-based access control techniques fail to detect this attack as those techniques fail to enforce or track the information control. The authors in [188] addressed this issue in the first place and divided their work into two subsections. The first subsection demonstrates a clear vision about the incompetency of the rule-based access mechanism. To generate the attack surface, they modeled a cross-app information flow graph that maps relations among apps through the shared control plane and starts actions on behalf of other applications; and the second section introduced ProvSDN, an online reference module that implements provenance graph tracks and record information flow. This module instrumented in ONOS consists of two modules: provenance collector: captures API call information. It identifies the association of incoming OpenFlow packets. The online reference monitor checks the current provenance graph concerning information flow in real-time and blocks the requests. The famous control planes like Floodlight, OpenDaylight do not provide a mechanism for such an issue. Thus, identification and opting mitigation mechanism as a mandatory property is crucial and left under future consideration.

6.6.3. Cross-protocol attack

In SDN, the communication processes are asynchronous and pass explicit text messages through the control channel during communications. Thus, SDN specification suggests the implementation of TLS in both the control plane-data plane communication channel and the control plane-application plane communication channel. However, TLS itself is suspected of a Cross-Protocol attack or Drown attack [189]. Therefore, even after the implementation of TLS in SDN, the threat will remain the same. The Drown attack can break the TLS or SSL encryption and steal the sensitive information of the network. Thus, in this respect, special consideration needs to be addressed. In recent work, Agborubere and Sanchez-Velazquez [107], an improvement in TCP handshaking can reduce the problem. In this research, the authors suggest that before the handshaking termination server must reverify the client status. Client, failing to do so terminates the handshake because of timestamp. This mechanism is the optimistic observation of the likelihood solution, but the rest is an open issue and under future consideration. In SDN, the communication processes are asynchronous, and explicit text messages pass through the control channel during communications. Thus, it is favorable to implement TLS in the control plane-data plane communication channel and control plane-application plane communication channel. However, TLS itself is suspected of a Cross-Protocol attack or Drown attack [189]. Therefore, even though SDN implements TLS, the threat will remain the same. The Drown attack can break the TLS or SSL encryption and steals the sensitive information of the network. Thus, in this respect, special consideration needs to be addressed. In recent work, Agborubere and Sanchez-Velazquez [107], an improvement in TCP handshaking can reduce the problem. In this research, the authors suggest that before the handshaking termination server must reverify the client status. Client failing to do so terminates the handshake because of timestamp. This mechanism draws an optimistic observation of the likelihood solution, but the rest is an open issue and under future consideration.

6.6.4. Reflection attack

This attack tricks the data plane events (direct or indirect) for which the control plane generates a massive amount of control messages and exhausts the data plane. The reflection attack takes the advantages of no or weak consistency and low availability constraint of SDN. This attack is more sophisticated and stealthier compared to direct DoS attacks. M. Zhang et al. introduced SWGuard [190] as a novel host-app abstraction mechanism that implements a priority-based assignment and scheduling algorithm to overcome the consequences of a reflection attack. Monitoring host or anomalous flow statistics is no longer useful in detecting reflection attacks as heterogeneous application-specific requirements can easily violate those mechanisms. Thus, SWGuard monitors host-application pair downlinks message anomalies and prioritizes downlink messages during channel congestion. The SWGuard successfully mitigated the Table-miss Striking Attack and Counter Manipulation Attack, two subclasses of a reflection attack. The prioritization mechanism of downlink messages leads to SWGuard capable of work under no adversary situations, i.e., attack under normal circumstances.

Moreover, Host-App abstraction helps SWGuard to recognize the switch port to identify a host. Thus, forging hosts is also not possible as for upstream messages, header fields are assigned by the switch hardware and uniquely identified during abstraction. However, this mechanism addresses only downlink messages. The uplink may be taken into account in the future to make this approach more convenient.

6.7. Section conclusion

This section includes 22 different attacks with enlightened background information on security loopholes in SDN layered architecture. It also discussed the existing mechanisms, components introduced to mitigate such attacks, and the existing mitigations. In this section, it is identified that many researchers extend the popular mitigation techniques to increase efficiency. TopoGuarad+ is an extension of Topoguard, LineSwitch is an extension of AVANT-GUARD, which increases efficiency of AVANT-GUARD. However, both AVANT-GUARD and LineSwitch fail to detect the ICMP packet floods. The Safeguard is a more recent scheme that works for TCP, SYN, and ICMP flooding. FloodSheild is also an extension of FloodDefender. Identifying limitations of existing works are also hectic and essential for the researcher. In order to reduce the same, this section addresses 25 different future extension propositions. Compared to other attack domains, it is identified that a few numbers of research works have been carried out in the field of Service Chain Jamming and newly identified attack such as Cross-Path attack, Cross-App, Cross-Protocol, Reflection attack. Similarly, only possible countermeasure for Reverse Loop and Topology Freezing attack are introduced in SDN architecture and practical implementation remain uncovered. This paper is also advantageous for the researchers to visualize the current trends in SDN information security. The security concerns are in current trends for the SDN architecture but not the only issue in SDN. The next section provides a glimpse of other issues that SDN suffers.

7. Other issues in SDN

The emergence of SDN can overcome the limitations of traditional networks, as addressed in Section 3. In Section 3, we address only the positive intent of scalability, QoS, and QoE concerning SDN environment implementation, but these three properties are also an open issue in SDN architecture and need special attention for further improvement.

7.1. Scalability

Segregating control plane with lower-level abstraction facilities and network-wide global view increases network management's potentiality compared to the traditional network. However, it faces a bottleneck with the emergent network events or sudden expansion of incoming network requests. The scalability of SDN resources has a direct impact on the availability issues in SDN.

A recent research work [191] identifies the flow request benchmark of existing control planes with the help of CBench [192] online tool. The outcome shows that the flow response rate of different available control planes like Beacon, ODL, and Maestro can reach up to 100 flows/ms; OpenMUL and Floodlight deal with 150 flows/ms. The control plane ONOS performs best among all others with maintaining a flow rate in the range of 400 flow/ms to 500 flow/ms and the famous control planes NOX, POX, and Ryu at the lowest. The capability of maintaining flow requests for each control plane is significantly well for the sizeable networking system, but it degrades with an extensive networking system like data centers.

To alleviate this concern, two approaches are suggested: (1) architectural approach and (2) mechanism related approach. The architectural approach suggests the placement of control planes in a distributed or hierarchical manner for maintaining sudden emergent of network events or sudden expansion of incoming network requests or any network failure [139–142]. The authors K. S. Atwal et al. hierarchically placed a control plane with maintaining a peer to peer communication in between the control plane to alleviate the scalability in the network [193].

The mechanism-based approach having two subcategories: Parallelism-based Optimization and Control Plane Routing Scheme-based Optimization. In the parallelism optimization control plane, ONOS outplays Beacon, ODL, and Maestro in the CBench test by providing flow management in the range of 400 flow/ms to 500 flow/ms. In routing scheme-based optimization, the authors in [194] introduced Flowcache, a software-based cache placed in between the control plane and the data plane. Flowcache capture updated flow information and can reduce the delay time of future similar transactions. In [195] authors introduced a routing and forwarding algorithm based on switch loadings, wired/wireless capacity, and flow characteristics. This algorithm minimizes the flow rule setup by providing the same routing path for all small duration flow entries.

7.2. Quality of service signaling

Under the subsection QoS of Section 3, we attend some crucial benefits of SDN compared to proving best-effort service, but this subsection addresses some critical issues of QoS in SDN architecture that required future improvement to achieve customer satisfaction.

SDN retains the control plane in a centralized point of the system and manages underneath topology utilizing control requests from the data plane. The control plane pushes the data plane to collect the flow statistics about the active flows. This mechanism helps the control plane to retain its global view but does not assure the quality of information processing. It is a burden for the control plane to achieve both simultaneously. The mechanism per-flow statistics maintenance might increase the granularity but suffers from scalability problems and reduces the system's quality of service. The FlowCover [196] is a low-cost monitoring scheme that globally optimizes the flow request by aggregating the incoming requests and forwarding replies. This scheme uses the greedy approach to select a targeted switch for flow rule installation based on all active polled flows instead of per-flow rule installation. Thus, it can reduce the 50% monitoring overhead of the system. The authors S. R. Choudhary et al. introduced PlayLess, a plug & play framework where a customized application can easily plug in the system [197]. It also provides a well-defined interface that can manage the application abstractions. It delivers dynamic flow statistics information by implementing an adaptive scheduling algorithm. This dynamism in collecting flow statistics does not reduce accuracy about flow information than the constant periodic polling method, thus reducing the system's extra monitoring overhead. Another prominent solution for this issue is OpenMeasure [198], smart network inference, and flow sampling framework residing on the SDN control plane. It uses an adaptive learning algorithm for learning flow statistics of the system. OpenMeasure improves the flow accuracy, optimizes resource allocation in the system, and reduces QoS signaling overhead.

7.3. Inter-domain QoS provisioning

Section 3 addresses the research work that focused on the provisioning of the QoS in intra-domain transaction measures. Most network transactions are conducted at the inter domain level between multiple autonomous networks, making inter-domain QoS as intra-domain QoS. However, such considerations are more challenging and remain future attention to achieve the SDN environment's absolute QoS abilities.

In this issue, the authors J. M. Wang et al. introduce a multi-class QoS-guaranteed traffic management scheme that supports joint bandwidth allocation to multiple flow classes and achieves the best bandwidth utilization in the system [199]. For conquering inter-domain traffic, a centralized traffic management server (TM server) is introduced like any other control plane and assigns the responsibility of coordinating the network activities. A bandwidth broker is responsible for collecting and aggregating the bandwidth demand of an application. Not only that, the bandwidth broker informs the aggregated demand to the TM server. An OpenFlow enabled switch informs the current network flow statistics to the TM server. The TM server computes the bandwidth allocation for the competing resources based on the collected information from bandwidth broker and OpenFlow enabled switch.

To solve the inter-domain QoS challenges, the authors in [200] introduce the Inter-Domain Routing Brokering Plane (IDRBP). This plane sticks to some predefined policies, which are: (1) providing a strict Service Layer Agreement (SLA) corresponds to QoS that support BGP. (2) an inter-domain routing learning model that can provide proactive service for the flow request and abstraction level. This mechanism successfully achieves network-level abstraction with scalable routing maintenance. The authors Bi et al. [201] manage the QoS in the network by separating control plane responsibility into two different segments. Intra-domain QoS management implements an information exchange signaling process in the southbound interface, and in the switch, it implements a Binding Cache Entry (BCE) table for signaling a handover process in the system. Moreover, for the inter-domain QoS, the inter-domain information exchange mechanism is employed in the east/west bound interface. By this information exchange, a control plane can achieve a global view of the inter-domain network. Not only that, it implements the Mobility Management Entity (MME) in the control plane east/westbound interface for which a control plane can co-locate other control planes in the inter-domain network.

7.4. Section conclusion

This section helps to cover all the aspects (positive or negative) of SDN. Each section poses multiple references and the best possible comparison to justify the stated statements in the particular section or subsections. The upcoming section addresses a few advanced research directions for which researchers can dive into convincing research propositions for the betterment of SDN architecture or SDN-based architecture.

8. Potential approaches for future research directions

The outcome of the extensive survey silently draws attention to Blockchain technology as a prominent solution to address issues. For example, multi-control plane architecture is introduced to maintain the availability of SDN networks. Still, it suffers from balancing the network load where Blockchain can be used to monitor the network traffic and estimate the load of each control plane and distribute the load. With the distribution of SDN in application setups together with cloud computing, IoT, and big data, the enormous requests of matched or unmatched traffic to the single SDN control plane can make the security and performance of the control plane a potential bottleneck of the SDN network. The SDN incorporating NFV or Blockchain technology can achieve potentiality to improve the QoS of SDN and the security of SDN. The SecSFT, DTARS, BlockAS, SILedger are suitable examples of recent research implementations of SDN with NFV or Blockchain technologies. The authors Ali et al. [202] introduce xDBAuth, a Blockchain-based cross-domain authentication and authorization framework for IoT networks, where the framework maintains a hierarchy of local and global smart contracts to perform the permission allocation and access control for internal and external user/IoT devices. A similar concept can be implemented in SDN (single or multi-control plane architecture) to resolve the highlighted issues.

Topological adversaries are possible in SDN architecture due to a lack of fairness in LLDP packets of network traffic. Those adversaries can mislead the SDN global view mentioned in earlier sections. There is a possibility for distributing the authentication mechanism of LLDP packets into the Blockchain and reducing such adversaries from the SDN network. Similarly, distributing the ARP authentication into the Blockchain can reduce the chances of DDoS attacks, Message Injection attacks, Man-in-Middle attacks. Not only that, multi-control plane or collaborative network where each control plane runs simultaneously is prone to several inter-domain or intra-domain attacks can be overcome by transferring authentication mechanism into the Blockchain.

SDN architecture is also prone to side-channel attacks because of its flow management mechanism. For an initial packet of a new traffic flow, SDN manages a unique packet known Packet_In message for validating the flow request and installing the flow rules into the flow table. Then, the rest of the traffic flows are directly sent to the next hop by mapping the existing flow rules of the flow table. This flow configuration delay creates a vulnerable surface for side-channel attacks. Transferring the validation mechanism or updating the status of flow rule installation into the Blockchain might reduce the chances of a side-channel attack because doing the same makes such attack points less for the intruder.

The smart-grid computing environment is responsible for automating, monitoring,and controlling the two-way flow of energy from power to plug. Incorporating IP-based communications in the smart grid can produce a unified network platform capable of interconnecting all devices. However, such communication isprone to DDoS and IP spoofing attacks. The author's [203] utilize the SDN global view and dynamic policy decision capability into the smart-grid network and produce a security risk model, mitigation policies, and end-to-end QoS for mitigating the possibilities of the mentioned attacks. Although the proposed model successfully achieves its target. However, multi-control plane architecture like SDN-Blockchain based Resource monitor or Threat detector, SDN-Blockchain with Role or Attribute-based resource or policy authenticator, Policy obfuscation can significantly reduce many attack scenarios and improve the QoS related services over the smart-grid computing environment.

Fog computing or Edge computing engages its nodes or fog nodes at the edge of the network to facilitate the operation of computing, i.e., provide on-demand access and deploy a cost-effective, secure, effective, and efficient computing environment. Each fog node implements its computing resources in order to achieve the goal. SDN can help the fog computing environment to achieve each node status or resource status and secure communications more efficiently than the traditional approach. Implementation of SDN-based policy management into the fog nodes can reduce the attack scenario of the fog computing environment. SDN with Blockchain can reduce the same more efficiently. SDN-Blockchain with Role-based resource authenticator, Role-based policy authenticator, Attribute-based resource authenticator, Attribute-based policy authenticator can be implemented in this environment to increase the security or QoS-related services of this domain.

SDN runs vendor-agnostic applications over the control plane. The problem with such applications is that two applications might cause rule conflicts during flow rule installation. The most devastating consequence is that those applications can destroy the entire SDN network by using bugs or API misuse. For overcoming such catastrophic implications in SDN, permission-based modeling or access control is popular in the SDN environment. However, those permission-based access controls do not represent generic or unified access permission for different available control planes. The research works carried by [204] and [205] rethink permission modeling and represent a unified permission modeling using NLP techniques. The NLP-based permission model takes API documents as input and produces synthesized permission. This NLP-based model helps network administrators to determine appropriate permission for the corresponding SDN network and assign permissions for each case. The NLP-based approach can escalate SDN research in the future. A similar approach can be incorporated in multi-control architecture might reduce availability, consistency, and QoS-related issues and make SDN architecture convenient for maximum adoption. In collaborative SDN network will also benefit similarly. SDN, Blockchain, and NLP-based approaches can enhance network security and help in deploying a generic smart contract in collaborative or multi-control plane architecture. Not only that, SDN-Blockchain with NLP-based resource or policy authenticator can be implemented in fog computing or smart grid computing environment to increase the security or QoS-related services of this domain.

The unique features global view and dynamic policy management of SDN attract networking domains such as cloud computing, IoT, Fog computing, grid computing. If intelligent and dynamic policy management replaces the present dynamic policy management, then inconsistencies in SDN architecture might be reduced convincingly. AI might be a suitable technology to achieve the goal in SDN architecture and encourage other networking domains to maximize the uses of SDN architecture as well.

The present research study tries to attend the existing information security flaws in the best possible way. However, to reach comprehensiveness addressing security testing tools is an essential aspect and improves future research directions. For elevating the security of SDN in practical scenarios, there are some works focused on making the tool to benchmark the security of SDN as follows:

Delta: The authors [206] represent Delta, a highly automated and inter-operable security tool to uncover unknown security problems in diverse SDN elements as well as SDN deployment. It implements a Blackbox fuzzification mechanism to discover unknown attack scenarios. The Blackbox fuzzification mechanism systematically explores the crucial data flow interchanges between SDN elements and discovers known and potentially unknown vulnerabilities.

AudiSDN: AudiSDN is another benchmarking security tool for recognizing flow inconsistency in various applications and abstraction layers of the SDN stack. The authors [207] also utilize the fuzzy approach and introduce a dedicated network policy fuzz-testing module that can detect malformed SDN policies from the state transaction diagram of flow rules.

RE-CHECKER: RESTful services implement programmable network services in SDN architecture. The programmable feature of these services is critically susceptible to employing adversaries in the network. The authors [208] introduce RE-CHECKER to distinguish vulnerability and bugs in RESTful services. This tool successfully identifies the design flaws in the network due to bugs in RESTful services and has the potential for restricting devastating consequences in the SDN. RE-CHECKER also implements Blackbox fuzzy mechanism to achieve the goal.

According to research findings, Delta, AudiSDN, and RE-CHECKER are the most recent, prominent, and successful security benchmarking tools. The prime concern of AudiSDN is to detect the flow level inconsistency, RE-CHECKER is to vulnerabilities and Bugs in RESTful services consider flow level inconsistencies, and Delta identifies policy inconsistency as well as to detect the known and unknown attacks in SDN stacks. Thus, Delta has the larger domain in security benchmarking. All three mechanisms implement fuzzy-based mechanisms and concentrate on flow level inconsistency to achieve their respective goal. In the future, the fuzzy-based mechanism can be upgraded by an AI mechanism to which a smart and intelligent detection mechanism can be possible. Not only that, the fuzzy implementation can be modified with an advanced fuzzy system like Pythagorean fuzzy. By implementing the advanced fuzzy system, the existing mechanism can achieve a higher order of inconsistency detection and improve efficiency and accuracy.

Due to the scope of this research, we leave the implementation details and left responsibility for the researchers to dive into the investigation of the mentioned future directions and purpose persistent solutions in the SDN domain or SDN-based Networking domain.

Therefore, the outcome of an extensive survey creates a complete and deep understating of information security-related issues and also pinpoints many future scopes in this domain. The upcoming section concludes the discussion of this paper.

9. Conclusion

The research study presents evidence for three sides of the security pyramid that SDN possesses in its architecture. One side of the security pyramid consists of the advancement it brings into the networking world compared to traditional network architecture. The other two sides consist of inherited vulnerability and its consequences on information security. This study also addresses the attacks that grab the advantages of weak information security in SDN architecture.

SDN brings promising improvements compared to traditional network architecture but is still not widely adopted by the network programmer or organization as SDN components possess vulnerability that creates loopholes in information security. Our research work tries to address all these issues along with currently available countermeasures. The vulnerabilities of components create loopholes in information security and thus also let the door open for attack generation. The discussion is not limited to vulnerability and information security loopholes but also attends to the SDN threats that are only achievable due to the incompetence of SDN components and their properties.

Undoubtedly the SDN mechanisms make network management more flexible, adaptable, and independent. The traditional network suffers from the rigidity of network resources. The SDN properties can overcome such incompetency and also be able to provide good quality services to end-users. The present contribution describes the positive intents of availability, QoS, and QoE of SDN in Section 3.2, but these three properties are also an open issue in SDN architecture and need special attention for further improvement. Although these three issues are not an actual course of our contribution, Section 7 addresses the concise illustration of these issues due to unfolding the maximum issues of SDN and trying to be aware of the researchers' utmost aspect within this comprehensive survey. More importantly, research findings identify 103 different possible future works in SDN architecture, and on top of that, it suggests some constructive ideas using new technologies that can help increase the promises of SDN.

This paper intends to represent a comprehensive idea of SDN, and by addressing all these issues, we successfully achieve our goal.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. Jain, S. Paul, Network virtualization and software defined networking for cloud computing: A survey, IEEE Commun. Mag. 51 (11) (2013) 24–31.
[2] S. Hernan, S. Lambert, T. Ostwald, A. Shostackt, Uncover security design flaws using the STRIDE approach, in: MSDN Magazine, Micerosoft, 2006, URL: https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach.
[3] D. Kreutz, F.M. Ramos, P. Verissimo, Towards secure and dependable software-defined networks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, in: HotSDN '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 55–60.
[4] C. Yoon, S. Lee, H. Kang, T. Park, S. Shin, V. Yegneswaran, P. Porras, G. Gu, Flow wars: Systemizing the attack surface and defenses in software-defined networks, IEEE/ACM Trans. Netw. 25 (6) (2017) 3514–3530.
[5] J. Chica, J. Cuatindioy, J. Botero, Security in SDN: A comprehensive survey, J. Netw. Comput. Appl. 159 (2020).
[6] E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446, RFC Editor, 2018.
[7] R. Klöti, V. Kotronis, P. Smith, OpenFlow: A security analysis, in: 2013 21st IEEE International Conference on Network Protocols (ICNP), 2013, pp. 1–6.
[8] D. Kreutz, F.M.V. Ramos, P.E. Veríssimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76.
[9] B.A.A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: Past, present, and future of programmable networks, IEEE Commun. Surv. Tutor. 16 (3) (2014) 1617–1634.
[10] Y. Jarraya, T. Madi, M. Debbabi, A survey and a layered taxonomy of software-defined networking, IEEE Commun. Surv. Tutor. 16 (4) (2014) 1955–1980.
[11] S. Scott-Hayward, S. Natarajan, S. Sezer, A survey of security in software defined networks, IEEE Commun. Surv. Tutor. 18 (1) (2016) 623–654.
[12] S. Scott-Hayward, G. O'Callaghan, S. Sezer, Sdn security: A survey, in: 2013 IEEE SDN for Future Networks and Services (SDN4FNS), 2013, pp. 1–7.
[13] Z. Shu, J. Wan, D. Li, J. Lin, A. Vasilakos, M. Imran, Security in software-defined networking: Threats and countermeasures, Mob. Netw. Appl. 21 (2016).
[14] A. Akhunzada, E. Ahmed, A. Gani, M.K. Khan, M. Imran, S. Guizani, Securing software defined networks: taxonomy, requirements, and open issues, IEEE Commun. Mag. 53 (4) (2015) 36–44.
[15] S. Dong, K. Abbas, R. Jain, A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments, IEEE Access 7 (2019) 80813–80828.
[16] I. Mathebula, B. Isong, N. Gasela, A.M. Abu-Mahfouz, Analysis of SDN-based security challenges and solution approaches for SDWSN usage, in: 2019 IEEE 28th International Symposium on Industrial Electronics (ISIE), 2019, pp. 1288–1293.
[17] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, J. van der Merwe, Design and implementation of a routing control platform, in: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2, in: NSDI'05, USENIX Association, USA, 2005, pp. 15–28.
[18] A. Greenberg, G. Hjalmtysson, D.A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, H. Zhang, A clean slate 4D approach to network control and management, SIGCOMM Comput. Commun. Rev. 35 (5) (2005) 41–54.
[19] M. Casado, T. Garfinkel, A. Akella, M. Freedman, D. Boneh, N. McKeown, S. Shenker, SANE: a protection architecture for enterprise networks, 2006.
[20] M. Casado, M.J. Freedman, J. Pettit, J. Luo, N. McKeown, S. Shenker, Ethane: Taking control of the enterprise, SIGCOMM Comput. Commun. Rev. 37 (4) (2007) 1–12.
[21] ONF Group, OpenFlow gives malware a caning, 2013, online. URL: https://www.opennetworking.org/wp-content/uploads/2013/02/cs-of.pdf.
[22] S. Hong, R. Baykov, L. Xu, S. Nadimpalli, G. Gu, Towards SDN-defined programmable BYOD (bring your own device) security, in: NDSS, 2016.
[23] C. Sieber, A. Blenk, D. Hock, M. Scheib, T. Höhn, S. Köhler, W. Kellerer, Network configuration with quality of service abstractions for SDN and legacy networks, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 1135–1136.
[24] S. Tomovic, I. Radusinovic, N. Prasad, Performance comparison of QoS routing algorithms applicable to large-scale SDN networks, in: IEEE EUROCON 2015 - International Conference on Computer As a Tool (EUROCON), 2015, pp. 1–6.
[25] M. Feng, S. Mao, T. Jiang, Enhancing the performance of future wireless networks with software defined networking, Springer Front. Inf. Technol. Electron. Eng. J. 17 (2016).

[26] P. Manzanares-Lopez, J. Malgosa-Sanahuja, J.P. Muñoz-Gea, A software-defined networking framework to provide dynamic QoS management in IEEE 802.11 networks, Sensors (Basel, Switzerland) 18 (2018).

[27] A. Hammad, A. Aguado, K. Kondepu, Y. Zong, J. Marhuenda, S. Yan, R. Nejabati, D. Simeonidou, Demonstration of NFV content delivery using SDN-enabled virtual infrastructures, in: 2017 Optical Fiber Communications Conference and Exhibition (OFC), 2017, pp. 1–2.

[28] R. Schatz, S. Schwarzmann, T. Zinner, O. Dobrijevic, E. Liotou, P. Počta, S. Barakovic, J. Barakovic Husic, L. Skorin-Kapov, QoE management for future networks, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2018, pp. 49–80.

[29] A. Ben Letaifa, G. Maher, S. Mouna, ML based QoE enhancement in SDN context: Video streaming case, in: 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), 2017, pp. 103–108.

[30] A. Letaifa, An adaptive machine learning-based QoE approach in SDN context for video-streaming services, Turk. J. Electr. Eng. Comput. Sci. 26 (2018) 2859–2871.

[31] N. Lawrence, SDN and its Role in Automating & Scaling in the Data Center (Master's thesis), IT and Computing. The Open University at UK, 2018.

[32] S. Bera, S. Misra, A.V. Vasilakos, Software-defined networking for internet of things: A survey, IEEE Internet Things J. 4 (6) (2017) 1994–2008.

[33] L. Shif, F. Wang, C. Lung, Improvement of security and scalability for IoT network using SD-VPN, in: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1–5.

[34] J. Romero Gázquez, M. Delgado, Software architecture solution based on SDN for an industrial IoT scenario, Wirel. Commun. Mob. Comput. 2018 (2018) 12.

[35] S.K. Tayyaba, M.A. Shah, O.A. Khan, A.W. Ahmed, Software defined network (SDN) based internet of things (IoT): A road ahead, in: Proceedings of the International Conference on Future Networks and Distributed Systems, in: ICFNDS '17, Association for Computing Machinery, New York, NY, USA, 2017.

[36] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, A. Vahdat, B4: Experience with a globally-deployed software defined wan, in: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, in: SIGCOMM '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 3–14.

[37] A. Vahdat, D. Clark, J. Rexford, A purpose-built global network: Google's move to SDN, Queue 13 (8) (2015) 100–125.

[38] J.-L. Luo, S.-Z. Yu, S.-J. Peng, SDN/NFV-based security service function tree for cloud, IEEE Access 8 (2020) 38538–38545.

[39] A. Zarca, M. Bagaa, J. Bernal Bernabe, T. Taleb, A. Skarmeta, Semantic-aware security orchestration in SDN/NFV-Enabled IoT systems, Sensors 20 (2020) 3622.

[40] P. Krishnan, S. Duttagupta, K. Achuthan, SDN/NFV security framework for fog-to-things computing infrastructure, Softw. - Pract. Exp. 50 (2019).

[41] C. Bouras, P. Ntarzanos, A. Papazois, Cost modeling for SDN/NFV based mobile 5G networks, in: 2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2016, pp. 56–61.

[42] B. Naudts, M. Kind, F. Westphal, S. Verbrugge, D. Colle, M. Pickavet, Techno-economic analysis of software defined networking as architecture for the virtualization of a mobile network, in: 2012 European Workshop on Software Defined Networking, 2012, pp. 67–72.

[43] E. Hernandez-Valencia, S. Izzo, B. Polonsky, How will NFV/SDN transform service provider opex? IEEE Netw. 29 (3) (2015) 60–67.

[44] C. Grossner, Data Center SDN Strategies Service Provider Survey, Technical Report, IHS Markit, North America, 2017.

[45] K.K. Karmakar, V. Varadharajan, U. Tupakula, On the design and implementation of a security architecture for software defined networks, in: 2016 IEEE 18th International Conference on High Performance Computing and Communications, 2016, pp. 671–678.

[46] S. Lee, C. Yoon, C. Lee, S. Shin, V. Yegneswaran, P.A. Porras, DELTA: A security assessment framework for software-defined networks, in: NDSS, 2017.

[47] J. Sonchack, A. Aviv, E. Keller, J. Smith, Enabling practical software-defined networking security applications with OFX, 2016.

[48] K.K. Karmakar, V. Varadharajan, U. Tupakula, Mitigating attacks in software defined network (SDN), in: 2017 Fourth International Conference on Software Defined Systems (SDS), 2017, pp. 112–117.

[49] R. Deb, S. Roy, Common vulnerability scoring system for SDN environment, Int. J. Eng. Adv. Technol. 8 (6) (2019) 4022–4029.

[50] R. Deb, S. Roy, Dynamic vulnerability assessments of software-defined networks, Innov. Syst. Softw. Eng. (2019) 1–7.

[51] P. Porras, S. Cheung, M. Fong, K. Skinner, V. Yegneswaran, Securing the software defined network control layer, 2015.

[52] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, G. Gu, A security enforcement kernel for OpenFlow networks, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, in: HotSDN '12, Association for Computing Machinery, New York, NY, USA, 2012, pp. 121–126.

[53] F. Klaedtke, G.O. Karame, R. Bifulco, H. Cui, Access control for SDN controllers, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, in: HotSDN '14, Association for Computing Machinery, 2014, pp. 219–220.

[54] M. Monaco, O. Michel, E. Keller, Applying operating system principles to SDN controller design, in: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, in: HotNets-XII, Association for Computing Machinery, 2013.

[55] H. Hoang, D. Phan, V.-H. Pham, A security-enhanced monitoring system for northbound interface in SDN using blockchain, in: SoICT 2019, 2019, pp. 197–204.

[56] W. Ren, Y. Sun, H. Luo, M. Guizani, SILedger: A blockchain and ABE-based access control for applications in SDN-IoT networks, IEEE Trans. Netw. Serv. Manag. (2021) 1.

[57] J.W. Kang, S.H. Park, J. You, Mynah: Enabling lightweight data plane authentication for SDN controllers, in: 2015 24th International Conference on Computer Communication and Networks (ICCCN), 2015, pp. 1–6.

[58] HP-Team, Hp Switch Software-Openflow Supplement, Technical Report, Hewlett-Packard Company, Roseville, California, 2012.

[59] A.V. Atli, M.S. Uluderya, S. Tatlicioglu, B. Gorkemli, A.M. Balci, Protecting SDN controller with per-flow buffering inside OpenFlow switches, in: 2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), 2017, pp. 1–5.

[60] H. Wang, L. Xu, G. Gu, FloodGuard: A DoS attack prevention extension in software-defined networks, in: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015, pp. 239–250.

[61] S. Gao, Z. Li, B. Xiao, G. Wei, Security threats in the data plane of software-defined networks, IEEE Netw. 32 (4) (2018) 108–113.

[62] P. Zhang, Towards rule enforcement verification for software defined networks, in: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1–9.

[63] J. Cao, M. Xu, q. li, K. Sun, Y. Yang, J. Zheng, Disrupting SDN via the data plane: A low-rate flow table overflow attack, 2018.

[64] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, J. Shen, Defending against flow table overloading attack in software-defined networks, IEEE Trans. Serv. Comput. 12 (2) (2019) 231–246.

[65] I. Abdulqadder, D. Zou, I. Aziz, B. Yuan, Validating user flows to protect software defined network environments, Secur. Commun. Netw. 2018 (2018) 1–14.

[66] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, S. Whyte, Real time network policy checking using header space analysis, in: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, in: nsdi'13, USENIX Association, USA, 2013, pp. 99–112.

[67] A. Khurshid, W. Zhou, M. Caesar, P. Godfrey, VeriFlow: Verifying network-wide invariants in real time, ACM SIGCOMM Comput. Commun. Rev. 42 (2012).

[68] N. Li, C. Yan, X. Wang, C. Wang, Conflict-aware network state updates in SDN, in: 2016 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), 2016, pp. 214–221.

[69] H. Hu, W. Han, G.-J. Ahn, Z. Zhao, FLOWGUARD: Building robust firewalls for software-defined networks, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, in: HotSDN '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 97–102.

[70] P. Wang, L. Huang, H. Xu, B. Leng, H. Guo, Rule anomalies detecting and resolving for software defined networks, in: 2015 IEEE Global Communications Conference (GLOBECOM), 2015, pp. 1–6.

[71] W. Zhou, D. Jin, J. Croft, M. Caesar, B. Godfrey, Enforcing customizable consistency properties in software-defined networks, in: NSDI, 2015.

[72] Q. Li, Y. Chen, P.P.C. Lee, M. Xu, K. Ren, Security policy violations in SDN data plane, IEEE/ACM Trans. Netw. 26 (4) (2018) 1715–1727.

[73] J. Cui, S. Zhou, H. Zhong, Y. Xu, K. Sha, Transaction-based flow rule conflict detection and resolution in SDN, in: 2018 27th International Conference on Computer Communication and Networks (ICCCN), 2018, pp. 1–9.

[74] A. Al-Najjar, S. Layeghy, M. Portmann, Pushing SDN to the end-host, network load balancing using OpenFlow, in: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), 2016, pp. 1–6.

[75] L. Xu, J. Huang, S. Hong, J. Zhang, G. Gu, Attacking the brain: Races in the SDN control plane, in: 26th USENIX Conference on Security Symposium, in: SEC'17, USENIX Association, USA, 2017, pp. 451–468.

[76] B.K. Tripathy, D.P. Das, S.K. Jena, P. Bera, Risk based security enforcement in software defined network, Comput. Secur. 78 (2018) 321–335.

[77] A. Akhunzada, A. Gani, N. Anuar, A. Aziz, K. Khan, A. Hayat, S. Khan, Secure and dependable software defined networks, J. Netw. Comput. Appl. (2015).

[78] H. Farhady, A. Nakao, Software-defined networking: A survey, Comput. Netw. 81 (2015).

[79] S. Hong, L. Xu, H. Wang, G. Gu, Poisoning network visibility in software-defined networks: New attacks and countermeasures, in: NDSS, 2015.

[80] M. Dhawan, R. Poddar, K. Mahajan, V. Mann, SPHINX: Detecting security attacks in software-defined networks, 2015.

[81] N. Kaur, A. Singh, N. Kumar, S. Srivastava, Performance impact of topology poisoning attack in SDN and its countermeasure, 2017, pp. 179–184.

[82] T. Alharbi, M. Portmann, F. Pakzad, The (in)security of topology discovery in openflow-based software defined network, Int. J. Netw. Secur. Appl. 10 (2018) 01–16.

[83] T. Alharbi, M. Portmann, F. Pakzad, The (in)security of topology discovery in software defined networks, in: 2015 IEEE 40th Conference on Local Computer Networks (LCN), 2015, pp. 502–505.

[84] M. Berman, J. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Seskar, GENI: A federated testbed for innovative network experiments, Comput. Netw. 61 (2014).

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

[86] R. Skowyra, L. Xu, G. Gu, V. Dedhia, T. Hobson, H. Okhravi, J. Landry, Effective topology tampering attacks and defenses in software-defined networks, in: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2018, pp. 374–385.

[87] C.V. Neu, A.F. Zorzo, A.M.S. Orozco, R.A. Michelin, An approach for detecting encrypted insider attacks on OpenFlow SDN networks, in: 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), 2016, pp. 210–215.

[88] V. Varadharajan, K. Karmakar, U. Tupakula, M. Hitchens, A policy-based security architecture for software-defined networks, IEEE Trans. Inf. Forensics Secur. 14 (4) (2019) 897–912.

[89] S. Liu, M.K. Reiter, V. Sekar, Flow reconnaissance via timing attacks on SDN switches, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 196–206.

[90] J. Sonchack, A.J. Aviv, E. Keller, Timing SDN control planes to infer network configurations, in: Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, Association for Computing Machinery, New York, NY, USA, 2016, pp. 19–22.

[91] J. Sonchack, A. Dubey, A.J. Aviv, J.M. Smith, E. Keller, Timing-based reconnaissance and defense in software-defined networks, in: Proceedings of the 32nd Annual Conference on Computer Security Applications, Association for Computing Machinery, New York, NY, USA, 2016, pp. 89–100.

[92] M. Conti, F.D. Gaspari, L.V. Mancini, Know your enemy: Stealth configuration-information gathering in SDN, 2016, CoRR abs/1608.04766.

[93] S. Khorsandroo, A.S. Tosun, Time inference attacks on software defined networks: Challenges and countermeasures, in: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 342–349.

[94] W. Wu, R. Liu, W. Ni, D. Kaafar, X. Huang, AC-PROT: An access control model to improve software-defined networking security, in: 2017 IEEE 85th Vehicular Technology Conference (VTC Spring), 2017, pp. 1–5.

[95] S. Scott-Hayward, C. Kane, S. Sezer, OperationCheckpoint: SDN application control, in: 2014 IEEE 22nd International Conference on Network Protocols, 2014, pp. 618–623.

[96] D. Menezes, O.C.M.B. Duarte, AuthFlow: authentication and access control mechanism for software defined networking, Ann. Telecommun. 71 (2016).

[97] J.Y. Cho, T. Szyrkowiec, Practical authentication and access control for software-defined networking over optical networks, in: Proceedings of the 2018 Workshop on Security in Softwarized Networks: Prospects and Challenges, in: SecSoN '18, Association for Computing Machinery, 2018, pp. 8–13.

[98] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, B.B. Kang, Rosemary: A robust, secure, and high-performance network operating system, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, in: CCS '14, Association for Computing Machinery, 2014, pp. 78–89.

[99] B. Chandrasekaran, B. Tschaen, T. Benson, Isolating and tolerating SDN application failures with LegoSDN, in: Proceedings of the Symposium on SDN Research, in: SOSR '16, Association for Computing Machinery, 2016.

[100] H. Cui, Z. Chen, L. Yu, K. Xie, Z. Xia, Authentication mechanism for network applications in SDN environments, in: 2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC), 2017, pp. 1–5.

[101] A.L. Aliyu, P. Bull, A. Abdallah, A trust management framework for network applications within an SDN environment, in: 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2017, pp. 93–98.

[102] K. Benzekki, A. ElundefinedFergougui, A. ElundefinedBelrhitiundefinedElundefinedAlaoui, Devolving IEEE 802.1X authentication capability to data plane in software-defined networking SDN architecture, Secur. Commun. Netw. 9 (17) (2016) 4369–4377, http://dx.doi.org/10.1002/sec.1613.

[103] H. Cheng, J. Liu, J. Mao, M. Wang, J. Chen, J. Bian, A compatible OpenFlow platform for enabling security enhancement in SDN, Secur. Commun. Netw. 2018 (2018) 1–20.

[104] A. Alshahrani, K. Suwais, B. Alkasasbeh, Authentication method in software-defined network based on ciphertext-policy attributes encryption, Int. J. Innovative Comput. Inf. Control 14 (2018) 1947–1957.

[105] F. Nife, Z. Kotulski, O. Reyad, New SDN-oriented distributed network security system, Appl. Math. Inf. Sci. 12 (2018) 673–683.

[106] O. Levillain, B. Gourdin, H. Debar, TLS record protocol: Security analysis and defense-in-depth countermeasures for HTTPS, 2015.

[107] B. Agborubere, E. Sanchez-Velazquez, OpenFlow communications and TLS security in software-defined networks, in: 2017 IEEE International Conference on Internet of Things (iThings), 2017, pp. 560–566.

[108] T. Mahboob, I. Arshad, A. Batool, M. Nawaz, Authentication mechanism to secure communication between wireless SDN planes, in: 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 2019, pp. 582–588.

[109] A.D. Ferguson, A. Guha, C. Liang, R. Fonseca, S. Krishnamurthi, Participatory networking: An API for application control of SDNs, SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 327–338.

[110] T. Sasaki, A. Perrig, D.E. Asoni, Control-plane isolation and recovery for a secure SDN architecture, in: 2016 IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 459–464.

[111] K. Lee, C. Lee, C. Hong, H. Yoo, Enhancing the isolation and performance of control planes for fog computing, Sensors (Switzerland) 18 (10) (2018).

[112] A. Dwaraki, S. Seetharaman, S. Natarajan, T. Wolf, State abstraction and management in software-defined networks, in: 2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2015, pp. 189–190.

[113] D. Dave, A. Nagaraju, A pragmatic analysis of security and integrity in software defined networks, 2017, pp. 733–740.

[114] L. Jacquin, A.L. Shaw, C. Dalton, Towards trusted software-defined networks using a hardware-based Integrity Measurement Architecture, in: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–6.

[115] D. Girtler, N. Paladi, Component integrity guarantees in software-defined networking infrastructure, in: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 292–296.

[116] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, A. Moore, OFLOPS: An open framework for OpenFlow switch evaluation, 2012.

[117] M. Kuzniar, P. Peresini, D. Kostic, Providing reliable FIB update acknowledgments in SDN, 2014, pp. 415–422, http://dx.doi.org/10.1145/2674005.2675006.

[118] K. Bu, X. Wen, B. Yang, Y. Chen, L.E. Li, X. Chen, Is every flow on the right track?: Inspect SDN forwarding with RuleScope, in: The 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9.

[119] A. Shukla, S. Schmid, A. Feldmann, A. Ludwig, S. Dudycz, A. Schuetze, Towards transiently secure updates in asynchronous SDNs, in: Proceedings of the 2016 ACM SIGCOMM Conference, in: SIGCOMM '16, Association for Computing Machinery, 2016, pp. 597–598.

[120] P.E. Heegaard, B.E. Helvik, V.B. Mendiratta, Achieving dependability in software-defined networking — A perspective, in: 2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM), 2015, pp. 63–70.

[121] P. Zhang, H. Li, C. Hu, L. Hu, L. Xiong, R. Wang, Y. Zhang, Mind the gap: Monitoring the control-data plane consistency in software defined networks, in: Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies, in: CoNEXT '16, Association for Computing Machinery, 2016, pp. 19–33.

[122] K. Lei, K. Li, J. Huang, W. Li, J. Xing, Y. Wang, Measuring the control-data plane consistency in software defined networking, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–7.

[123] I. Maity, A. Mondal, S. Misra, C. Mandal, CURE: Consistent update with redundancy reduction in SDN, IEEE Trans. Commun. 66 (9) (2018) 3974–3981.

[124] T.D. Nguyen, M. Chiesa, M. Canini, Decentralized consistent updates in SDN, in: Proceedings of the Symposium on SDN Research, in: SOSR '17, Association for Computing Machinery, 2017, pp. 21–33.

[125] K. Wu, J. Liang, S. Lee, Y. Tseng, Efficient and consistent flow update for software defined networks, IEEE J. Sel. Areas Commun. 36 (3) (2018) 411–421.

[126] F. Benamrane, M. Ben Mamoun, B. Redouane, New method for controller-to-controller communication in distributed SDN architecture, Int. J. Commun. Netw. Distrib. Syst. 19 (2017) 357.

[127] A.S. Muqaddas, P. Giaccone, A. Bianco, G. Maier, Inter-controller traffic to support consistency in ONOS clusters, IEEE Trans. Netw. Serv. Manag. 14 (4) (2017) 1018–1031.

[128] B. Halder, M.S. Barik, C. Mazumdar, Detection of flow violation in distributed SDN controller, in: 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), 2018, pp. 1–6.

[129] Y. Lu, Q. Fu, X. Xi, Z. Chen, E. Zou, B. Fu, A policy conflict detection mechanism for multi-controller software-defined networks, Int. J. Distrib. Sens. Netw. 15 (2019) 155014771984471.

[130] F. Bannour, S. Souihi, A. Mellouk, Adaptive state consistency for distributed ONOS controllers, in: 2018 IEEE Global Communications Conference (GLOBECOM), 2018, pp. 1–7.

[131] E. Sakic, W. Kellerer, Impact of adaptive consistency on distributed SDN applications: An empirical study, IEEE J. Sel. Areas Commun. 36 (12) (2018) 2702–2715.

[132] A. Panda, W. Zheng, X. Hu, A. Krishnamurthy, S. Shenker, SCL: Simplifying distributed SDN control planes, in: Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation, in: NSDI'17, USENIX Association, USA, 2017, pp. 329–345.

[133] E. Sakic, F. Sardis, J.W. Guck, W. Kellerer, Towards adaptive state consistency in distributed SDN control plane, in: 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1–7.

[134] H. Wang, G. Yang, P. Chinprutthiwong, L. Xu, Y. Zhang, G. Gu, Towards Fine-Grained Network Security Forensics and Diagnosis in the SDN Era, in: CCS '18, ACM, 2018, pp. 3–16.

[135] F.A. Lopes, M. Santos, R. Fidalgo, S. Fernandes, Model-driven networking: A novel approach for SDN applications development, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 770–773.

[136] W. Wang, C. Liu, J. Su, W. He, Achieving consistent SDN control with declarative applications, in: Proceedings of the 2016 ACM SIGCOMM Conference, in: SIGCOMM '16, 2016, pp. 585–586.

[137] Y. Xu, Y. Liu, R. Singh, S. Tao, Identifying SDN state inconsistency in OpenStack, in: Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, in: SOSR '15, 2015.

[138] G. Nencioni, B.E. Helvik, A.J. Gonzalez, P.E. Heegaard, A. Kamisinski, Availability modelling of software-defined backbone networks, in: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), 2016, pp. 105–112.

[139] F. Longo, S. Distefano, D. Bruneo, M. Scarpa, Dependability modeling of software defined networking, Comput. Netw. 83 (C) (2015) 280–296.

[140] T.A. Nguyen, T. Eom, S. An, J. Park, J. Hong, D. Kim, Availability modeling and analysis for software defined networks, 2015, pp. 149–168.

[141] A.D. Ferguson, A. Guha, C. Liang, R. Fonseca, S. Krishnamurthi, Hierarchical policies for software defined networks, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, in: HotSDN '12, Association for Computing Machinery, New York, NY, USA, 2012, pp. 37–42.

[142] A. Krishnamurthy, S.P. Chandrabose, A. Gember-Jacobson, Pratyaastha: An efficient elastic distributed SDN control plane, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, in: HotSDN '14, Association for Computing Machinery, 2014, pp. 133–138.

[143] N. Paladi, C. Gehrmann, SDN access control for the masses, Comput. Secur. 80 (2018).

[144] M. Azab, J. Fortes, Towards proactive SDN-controller attack and failure resilience, 2017, pp. 442–448.

[145] F.J. Ros, P.M. Ruiz, Five nines of southbound reliability in software-defined networks, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, in: HotSDN '14, Association for Computing Machinery, 2014, pp. 31–36.

[146] P. Vizarreta, P. Heegaard, B. Helvik, W. Kellerer, C. Mas Machuca, Characterization of failure dynamics in SDN controllers, 2017, pp. 1–7.

[147] J. Cui, Q. Lu, H. Zhong, M. Tian, L. Liu, A load-balancing mechanism for distributed SDN control plane using response time, IEEE Trans. Netw. Serv. Manag. 15 (4) (2018) 1197–1206.

[148] P.T. Duy, D.T.T. Hien, H.P. Qui, V.-H. Pham, Aloba: A mechanism of adaptive load balancing and failure recovery in distributed SDN controllers, in: 2019 IEEE 19th International Conference on Communication Technology (ICCT), 2019, pp. 1322–1326, http://dx.doi.org/10.1109/ICCT46805.2019.8947182.

[149] H. Jo, J. Nam, S. Seungwon, NOSArmor: Building a secure network operating system, Secur. Commun. Netw. 2018 (2018) 1–14.

[150] Y. Zhou, K. Chen, J. Zhang, J. Leng, Y. Tang, Exploiting the vulnerability of flow table overflow in software-defined network: Attack model, evaluation, and defense, Secur. Commun. Netw. 2018 (2018) 1–15, http://dx.doi.org/10.1155/2018/4760632.

[151] S. Seungwon, V. Yegneswaran, P. Porras, G. Gu, AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks, 2013, http://dx.doi.org/10.1145/2508859.2516684.

[152] D. Yu, Authentication for resilience: The case of SDN (Transcript of discussion), in: Security Protocols Workshop, 2013.

[153] Y. Tseng, Z. Zhang, F. Naï t Abdesselam, ControllerSEPA: A security-enhancing SDN controller plug-in for OpenFlow applications, 2016, pp. 268–273, http://dx.doi.org/10.1109/PDCAT.2016.064.

[154] H. Li, P. Li, S. Guo, S. Yu, Byzantine-resilient secure software-defined networks with multiple controllers in cloud, IEEE Trans. Cloud Comput. 2 (2014) 695–700.

[155] H. Cheng, J. Liu, J. Mao, M. Wang, J. Chen, OSCO: An open security-enhanced compatible OpenFlow platform, 2018, pp. 66–77.

[156] S. Xiang, H. Zhu, L. Xiao, W. Xie, Modeling and verifying TopoGuard in OpenFlow-based software defined networks, in: 2018 International Symposium on Theoretical Aspects of Software Engineering (TASE), 2018, pp. 84–91.

[157] E. Marin, N. Bucciol, M. Conti, An in-depth look into SDN topology discovery mechanisms: Novel attacks and practical countermeasures, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, in: CCS '19, ACM, 2019, pp. 1101–1114.

[158] D. Kotani, Y. Okabe, A packet-in message filtering mechanism for protection of control plane in openflow networks, in: 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2014, pp. 29–40.

[159] M. Ambrosin, M. Conti, F. De Gaspari, R. Poovendran, LineSwitch: Tackling control plane saturation attacks in software-defined networking, IEEE/ACM Trans. Netw. 25 (2) (2017) 1206–1219.

[160] B. Han, X. Yang, Z. Sun, J. Huang, J. Su, OverWatch: A cross-plane ddos attack defense framework with collaborative intelligence in SDN, Secur. Commun. Netw. 2018 (2018) 1–15.

[161] Y. Wang, T. Hu, G. Tang, J. Xie, J. Lu, SGS: Safe-guard scheme for protecting control plane against ddos attacks in software-defined networking, IEEE Access 7 (2019) 34699–34710.

[162] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, R. Smeliansky, Advanced study of SDN/OpenFlow controllers, in: ACM Proc. CEE-SECR, 2013.

[163] S. Deng, X. Gao, Z. Lu, X. Gao, Packet injection attack and its defense in software-defined networks, IEEE Trans. Inf. Forensics Secur. 13 (3) (2018) 695–705.

[164] A.S. Alshra'a, J. Seitz, Using INSPECTOR device to stop packet injection attack in SDN, IEEE Commun. Lett. 23 (7) (2019) 1174–1177.

[165] C. Röpke, T. Holz, On network operating system security, 26 (2016) 6–24.

[166] J. Noh, S. Lee, J. Park, S. Seungwon, B. Kang, Vulnerabilities of network OS and mitigation with state-based permission system, Secur. Commun. Netw. 9 (2015) 1971–1982, http://dx.doi.org/10.1002/sec.1369.

[167] C. Röpke, T. Holz, Preventing malicious SDN applications from hiding adverse network manipulations, in: Proceedings of the 2018 Workshop on Security in Softwarized Networks: Prospects and Challenges, in: SecSoN '18, Association for Computing Machinery, 2018, pp. 40–45.

[168] C. Qi, J. Wu, H. Hu, G. Cheng, W. Liu, J. Ai, C. Yang, An intensive security architecture with multi-controller for SDN, in: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2016, pp. 401–402.

[169] C. Röpke, T. Holz, SDN rootkits: Subverting network operating systems of software-defined networks, 2015, pp. 339–356.

[170] C. Lee, C. Yoon, S. Shin, S.K. Cha, INDAGO: A new framework for detecting malicious SDN applications, in: 2018 IEEE 26th International Conference on Network Protocols (ICNP), 2018, pp. 220–230.

[171] X. Wen, Y. Chen, C. Hu, C. Shi, Y. Wang, Towards a secure controller platform for openflow applications, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, in: HotSDN '13, Association for Computing Machinery, 2013, pp. 171–172.

[172] X. Wen, Y. Chen, C. Hu, C. Shi, Y. Wang, Towards a secure controller platform for openflow applications, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13, New York, NY, USA, 2013, pp. 171–172.

[173] A.C. Lapolli, J. Adilson Marques, L.P. Gaspary, Offloading real-time DDoS attack detection to programmable data planes, in: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2019, pp. 19–27.

[174] S. Wang, S. Chandrasekharan, K. Gomez, S. Kandeepan, A. Al-Hourani, M.R. Asghar, G. Russello, P. Zanna, SECOD: SDN sEcure control and data plane algorithm for detecting and defending against DoS attacks, in: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1–5.

[175] M. Myint Oo, S. Kamolphiwong, T. Kamolphiwong, S. Vasupongayya, Advanced support vector machine-(ASVM-) based detection for distributed denial of service (DDoS) attack on software defined networking (SDN), J. Comput. Netw. Commun. 2019 (2019) 1–12.

[176] R. Bifulco, H. Cui, G.O. Karame, F. Klaedtke, Fingerprinting software-defined networks, in: 2015 IEEE 23rd International Conference on Network Protocols (ICNP), 2015, pp. 453–459.

[177] A. Azzouni, O. Braham, T.M. Trang Nguyen, G. Pujolle, R. Boutaba, Fingerprinting OpenFlow controllers: The first step to attack an SDN control plane, in: 2016 IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6.

[178] M. Zhang, J. Hou, Z. Zhang, W. Shi, B. Qin, B. Liang, Fine-grained fingerprinting threats to software-defined networks, in: 2017 IEEE Trustcom/BigDataSE/ICESS, 2017, pp. 128–135.

[179] Y. Qian, W. You, K. Qian, OpenFlow flow table overflow attacks and countermeasures, in: 2016 European Conference on Networks and Communications (EuCNC), 2016, pp. 205–209.

[180] T. Xu, D. Gao, P. Dong, C.H. Foh, H. Zhang, Mitigating the table-overflow attack in software-defined networking, IEEE Trans. Netw. Serv. Manag. 14 (4) (2017) 1086–1097.

[181] M. Brooks, B. Yang, A man-in-the-middle attack against OpenDayLight SDN controller, in: Proceedings of the 4th Annual ACM Conference on Research in Information Technology, in: RIIT '15, 2015, pp. 45–49.

[182] K. Zhang, X. Qiu, CMD: A convincing mechanism for MITM detection in SDN, in: 2018 IEEE International Conference on Consumer Electronics (ICCE), 2018, pp. 1–6.

[183] A. Aseeri, N. Netjinda, R. Hewett, Alleviating eavesdropping attacks in software-defined networking data plane, in: Proceedings of the 12th Annual Conference on Cyber and Information Security Research, in: CISRC '17, Association for Computing Machinery, 2017.

[184] G.K. Ndonda, R. Sadre, A low-delay SDN-based countermeasure to eavesdropping attacks in industrial control systems, in: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 1–7.

[185] G. Shang, P. Zhe, X. Bin, H. Aiqun, R. Kui, FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks, 2017, pp. 1–9, http://dx.doi.org/10.1109/INFOCOM.2017.8057009.

[186] M. Zhang, J. Bi, J. Bai, G. Li, FloodShield: Securing the SDN infrastructure against denial-of-service attacks, 2018, pp. 687–698.

[187] J. Cao, Q. Li, R. Xie, K. Sun, G. Gu, M. Xu, Y. Yang, The crosspath attack: Disrupting the SDN control channel via shared links, in: Proceedings of the 28th USENIX Conference on Security Symposium, in: SEC'19, USENIX Association, USA, 2019, pp. 19–36.

[188] B.E. Ujcich, S. Jero, A. Edmundson, Q. Wang, R. Skowyra, J. Landry, A. Bates, W.H. Sanders, C. Nita-Rotaru, H. Okhravi, Cross-app poisoning in software-defined networking, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, in: CCS '18, Association for Computing Machinery, 2018, pp. 648–663.

[189] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J.A. Halderman, V. Dukhovni, E. Käsper, S. Cohney, S. Engels, C. Paar, Y. Shavitt, DROWN: Breaking TLS using SSLv2, in: 25th USENIX Security Symposium (USENIX Security 16), USENIX Association, Austin, TX, 2016, pp. 689–706.

[190] M. Zhang, G. Li, L. Xu, J. Bi, G. Gu, J. Bai, Control Plane Reflection Attacks in SDNs: New Attacks and Countermeasures: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings, 2018, pp. 161–183.

[191] L. Zhu, M.M. Karim, K. Sharif, F. Li, X. Du, M. Guizani, SDN controllers: Benchmarking & performance evaluation, 2019, ArXiv abs/1902.04491.

[192] R. Sherwood, K.-K. Yap, Cbench controller benchmarker, 2011, URL: http://www.openflow.org/wk/index.php/Oflops.

[193] K.S. Atwal, A. Guleria, M. Bassiouni, A scalable peer-to-peer control plane architecture for software defined networks, in: 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA), 2016, pp. 148–152.

[194] A. Ruia, C. Casey, S. Saha, A. Sprintson, Flowcache: A cache-based approach for improving SDN scalability, 2016, pp. 610–615.

[195] C. Chuang, Y. Yu, A. Pang, Flow-aware routing and forwarding for SDN scalability in wireless data centers, IEEE Trans. Netw. Serv. Manag. 15 (4) (2018) 1676–1691.

[196] Z. Su, T. Wang, Y. Xia, M. Hamdi, FlowCover: Low-cost flow monitoring scheme in software defined networks, in: 2014 IEEE Global Communications Conference, GLOBECOM 2014, 2015, pp. 1956–1961.

[197] S. Chowdhury, M.F. Bari, R. Ahmed, R. Boutaba, PayLess: A low cost network monitoring framework for software defined networks, in: IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in A Software Defined World, 2014, pp. 1–9.

[198] C. Liu, A. Malboubi, C. Chuah, OpenMeasure: Adaptive flow measurement inference with online learning in SDN, in: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2016, pp. 47–52.

[199] Y. Bi, G. Han, C. Lin, M. Guizani, X. Wang, Mobility management for intro/inter domain handover in software-defined networks, IEEE J. Sel. Areas Commun. 37 (8) (2019) 1739–1754.

[200] J. Olorunfemi Abe, H. Ali Mantar, Multipath routing and brokering in inter-domain or inter-as with SDN: A model, in: 2017 Advances in Wireless and Optical Communications (RTUWO), 2017, pp. 192–197.

[201] Y. Bi, G. Han, C. Lin, M. Guizani, X. Wang, Mobility management for intro/inter domain handover in software-defined networks, IEEE J. Sel. Areas Commun. 37 (8) (2019) 1739–1754.

[202] G. Ali, N. Ahmad, Y. Cao, S. Khan, H. Cruickshank, E.A. Qazi, A. Ali, xDBAuth: Blockchain based cross domain authentication and authorization framework for internet of things, IEEE Access 8 (2020) 58800–58816.

[203] H. Maziku, S. Shetty, D. Nicol, Security risk assessment for sdn-enabled smart grids, Comput. Commun. 133 (2018).

[204] H. Kang, S. Seungwon, V. Yegneswaran, S. Ghosh, P. Porras, AEGIS: An Automated Permission Generation and Verification System for SDNs, 2018, pp. 20–26.

[205] H. Kang, V. Yegneswaran, S. Ghoshz, P. Porras, S. Seungwon, Automated permission model generation for securing sdn control-plane, IEEE Trans. Inform. Forensics Security PP (2019) 1–1.

[206] S. Lee, J. Kim, S. Woo, C. Yoon, S. Scott-Hayward, V. Yegneswaran, P. Porras, S. Seungwon, A comprehensive security assessment framework for software-defined networks, Comput. Security 91 (2020) 101720.

[207] S. Lee, S. Woo, J. Kim, V. Yegneswaran, P. Porras, S. Seungwon, AudiSDN: Automated Detection of Network Policy Inconsistencies in Software-Defined Networks, 2020, pp. 1788–1797.

[208] S. Woo, S. Lee, J. Kim, S. Shin, RE-CHECKER: Towards Secure RESTful Service in Software-Defined Networking, in: 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2018, pp. 1–5.

Raktim Deb received the MCA degree in Computer Application from the Assam Engineering College, Gauhati University, Guwahati, Assam, India, 2008, and the M. Tech. degree in Information Technology from Gauhati University in 2012. He is currently pursuing a Ph.D. degree with the Department of Computer Science and Engineering, Triguna Sen School of Technology, Assam University, Silchar, Assam, India, with research focusing on Software Defined Network security. He is a member of IEEE.

Sudipta Roy, Ph.D., is Professor Department of Computer Science and Engineering, Triguna Sen School of Technology, Assam University, Silchar, Assam. He was associated with ACE Consultants, Kolkata, as a Software Professional, and Bengal Institute of Technology, Kolkata, India, as a faculty. His research interests are image processing, soft computing techniques, and applications. He is presently guiding Ph.D. students, postgraduate and graduate students. He has published more than one hundred and fifty papers in international and national journals and conference proceedings. He is a member of FIETE, LMCSI, SMIEEE, MIE, MINNS, LMISTE.