International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 12, No.1 (Jul-2022)

https://dx.doi.org/10.12785/ijcds/120131

Software-Defined Networking Security Challenges and Solutions: A Comprehensive Survey

Pulkit Ohri[1] **and Subhrendu Guha Neogi**[2]

[1,2]*ASET, Amity University Madhya Pradesh, Gwalior, India*

Received 15 Aug.2021, Revised 22 Mar. 2022, Accepted 9 Jul. 2022, Published 20 July 2022

Abstract: Software-Defined Networking (SDN) has revolutionized the networking field by addressing scalability, flexibility, and control mechanism issues, which are present in Traditional Networks. One of the major advantages of SDN over Traditional Networks is centralized control over the network. In Traditional Networks, no separate Control Layer is present, due to which configuration and management of the network become extremely difficult. SDN introduces a separate Control Layer, which makes network management an easy and reliable process. Network management is now flexible and does not require expensive hardware and configuration changes. From this, not only the network administrator but also the network intruder benefits. SDN separates the Control and Data Plane of the Network. Due to the separation of the Control Plane, DoS/DDoS attacks on SDN Controller can cause a Single Point of Failure of the entire network, and hence security becomes vital for SDN. In this paper, the security challenges of all three layers of SDN, and solutions that need to be taken by the network administrator has been discussed. Various key benefits of SDN Security has been discussed in this paper. SDN security remains one of the hot topics in the field of networking, and our main goal remains to review the major security advancements, with their scope and limitations.

Keywords: Distributed Denial-of-Service, Open Network Operating System, OpenDaylight, Intrusion Detection System, Blockchain

1. Introduction

New era of Network-Design Paradigm has changed the way of communication, due to transformation from Traditional Network to Internet of Things (IoT) enabled network, and invention of low cost digital devices [1]. Traditional hardware-based networks are strongly coupled with the control and data plane. Due to diverse policies implemented in routers and switches, the devices in the network need to be controlled and SDN can help to control devices remotely. To make a configuration change in the network, devices need to be configured separately, which is a cumbersome task.

Software-Defined Networking (SDN) is a dynamic, programmable, and centrally controlled network management approach [2]. SDN solves the issues of Traditional networks, by separating the Control and Data Plane of a network. This heterogeneous approach makes networks agile and flexible. The North-Bound Application Programming Interface (API) Interface provides communication between SDN Controllers and applications of higher layer whereas South-Bound API provides interface between SDN Controller and other network nodes connected with lower level devices. By deployment of the programming interface, network administrators can now centrally control the network and can reinforce, update and manage network policies on the fly. Due to the separation of forwarding and data plane, management of SDN becomes flexible and easily configurable. Open Networking Foundation (ONF) started the separation of forwarding and data plane in 2011 [3]. OpenFlow protocol was standardized for communication between forwarding and control plane in 2012. Open Network Operating System (ONOS) released an open-source SDN Controller in 2014 [4]. To date, SDN has revolutionized the way Network administrators control their Networks. Using SDN, administrators can apply customization to devices on the fly rather than waiting for the vendors to apply these policies separately in the switches and routers. The major advantages of SDN are virtualization of the Network, lower operational costs, and reduced capital expenditures [5].

With the diverse advantages provided by the SDN, security is still a major concern. The separation of the forwarding plane causes SDN to get attacked by malware. Once the SDN Controller is compromised, the whole network consisting of switches, routers, and network hosts will go down. SDN suffers from a Single Point of Failure in the Control layer. This Single Point of Failure of SDN has raised a lot of concern among Network Administrators. Multiple issues are identified by the research community regarding SDN and diverse solutions are also provided for the same.

E-mail address: pulkit.ohri@s.amity.edu, sgneogi@gwa.amity.edu

This paper delves out various state-of-the-art SDN security paradigm and their recent trends with various developments proposed for the new era of network communication. The remaining part of the article has been organized as:

1. This paper consists of three sections. The first section discusses the architecture of SDN in depth. The second section discusses security issues present in each layer with their detailed impact on the SDN architecture. The major and state-of-art countermeasures proposed by the research community over a period of time are analyzed in detail with the discussion of their limitations.

2. The last section discusses the research gap present between the security issues and existing research. Open security challenges and solutions for the SDN security paradigm are helpful for future developments have been discussed at the end.

2. THREE-TIER ARCHITECTURE OF SDN

SDN separates programming from the underlying hardware and thus by programming on Control Layer information delivery is centralized. SDN follows three-layered architecture that includes Application, Control, and Data Layer [6]. SDN separates the routing decision from the Data plane to Control Plane. The Control plane decides from which path to transfer data and the data plane simply forwards the data through specific routers and switches.

A. Control Layer: The Brain of SDN

The Control layer is the brain of SDN and consists of SDN Controller for routing decisions. Different vendors have provided SDN Controllers and the major stakeholders are Open Daylight (ODL), Open Networking Operating System (ONOS), Network Operating System (NOX), Python Operating System (POX), Beacon, and Ryu. Among these SDN Controllers, ODL and ONOS are the most popular and trending ones in the market. ODL controller follows OpenFlow Protocol to cater to the need of the communication between Data and Control Plane of the SDN. Routing decisions are decided by the Control layer and the Data layer merely forwards the data to a specific port.

Separate SDN Controller provides Network Abstraction to the Network administrators. They can change policies using a programmable interface without worrying about the underlining hardware interface. South-Bound Application Programming Interface (API) acts as a bridge between SDN Controller and Data Plane. Communication between these two is done using OpenFlow Protocol. Using OpenFlow Protocol, the Controller can update flow table rules directly in the Data Plane OpenFlow Switch [7].

B. Data Infrastructure/Forwarding Layer

Data Layer, the bottom-most layer consists of switches that are known as OpenFlow Switches. OpenFlow Switches contain a flow table and a group table. OpenFlow Switches merely send the data to the destined port according to the flow table rules [8]. OpenFlow Switches only receive instructions from SDN Controller. Communication from SDN Controller to OpenFlow Switches is done securely using OpenFlow Protocol. OpenFlow Protocol uses Secure Sockets Layer (SSL) to preserve the integrity and confidentiality of data. When any Data Packet arrives at the OpenFlow Switch, it checks its flow tables to find a matching entry of IP Network. If the entry is found, it merely forwards the data to the destined port. If the entry is not found it sends the packet to the SDN Controller using the OpenFlow Protocol. The Controller will send an updated rule back to the OpenFlow Switch and then the Switch can forward the data to the destination port.

Traditional Switches are capable of both forwarding and routing decisions. OpenFlow Switch decouples these two and offers a variety of advantages. SDN Controller has now the high-level view of the routing path used and it can transfer data now to less utilized routes to increase Network speed and thus reduce network bottlenecks. Load Balancing, Traffic Management, Scalability are some of the other advantages of using OpenFlow Switches.

C. Application Layer

SDN Application layer, the topmost layer contains Network Applications for Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Firewall, and Cloud Orchestration. It also contains load Balancing, Policy Enforcement, Topology Viewer, Network Automation, and Management Application. Communication between the Application and Control Layer is done via North Bound Interface. Using the Application Layer, Network Administrators can program the SDN Controller to enforce new administrative policies and to control switches centrally. New Applications can be added by the Network Vendors in the Application layer by using Network statistics, Topology, and State.

North-Bound Application Programming Interface (API) acts as a bridge between SDN Applications and the Control Layer. Using Graphical User Interface Network administrator can access the North-Bound API. Using this Abstraction administrators can change network Policies on the fly without worrying about the underlying hardware. Detailed SDN Architecture has been depicted in Figure 1 containing the three layers: Application, Control, and Data Layer.

A packet originating from the Application layer is processed in the following manner. When a request packet arrives from an outside IP Address to the OpenFlow Switch, it checks its flow table entries for a match. If the match is found, the packet is sent to the specified destination port. If not, a PACKET-IN message is generated and the packet is sent to the SDN controller using South-Bound Interface for closer inspection. The controller now decides where to send the packet and it sends the new flow table rule back to the OpenFlow Switch. OpenFlow Switch now simply passes

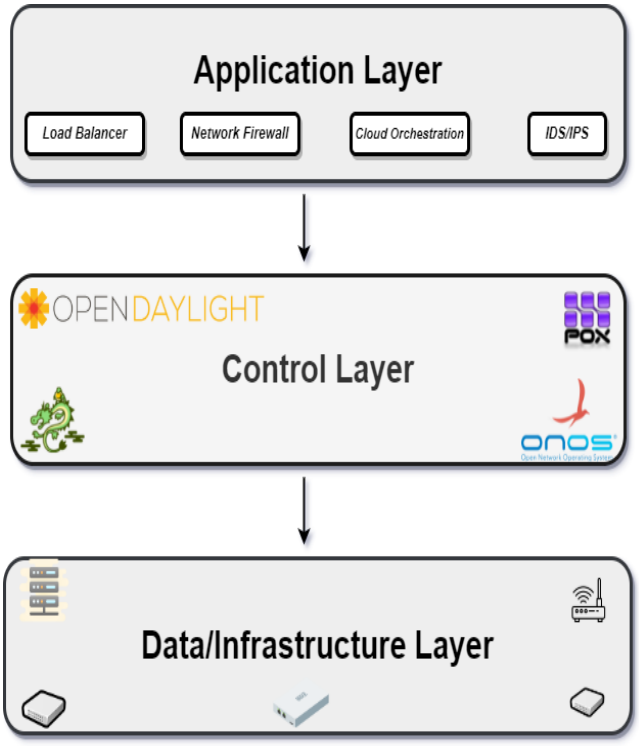

Figure 1. Three-Tier Architecture of SDN.

the packet to the destination port.

Decision-making lies solely in the hand of the SDN Controller and OpenFlow Switches simply becomes an information transfer device.

3. ADVANTAGES OF SDN OVER CLASSIC TRADITIONAL NETWORKS

SDN offers numerous advantages to the network administrator which was not possible with Traditional Networks. Traditional Networks works on hardware-based mechanisms and SDN works on software-based mechanism. The main advantage of SDN is the programmability provided by SDN. In Traditional Networks, networking was hardware-based and re-configuration of devices was not an easy task. Updating of devices requires a change of code in each and every router and switch. These are written in vendor-specific code, which is difficult to understand and update. But in SDN using simple programming, the network administrator can make changes in the entire network at will.

Traffic management is centralized and easily manageable since all traffic passes through the control layer. In such cases, a full map of the traffic flow is available to the network administrator. Benign and malicious applications can be easily identified by measuring the traffic flow and hence security becomes tightened. Error management and fault tolerance are high in SDN as compared to Traditional Networks. In Traditional Networks, due to distributed control identifying a faulty router or switch is difficult and a time-consuming task. In SDN, due to programming, it is an easy and automatic process to identify faulty devices.

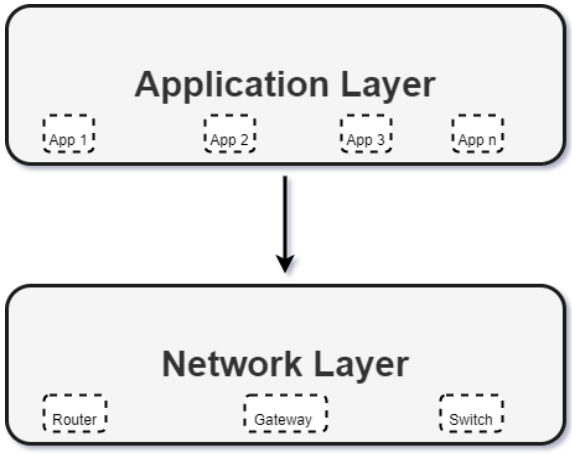

Figure 2. Communication Layers of the Traditional Network. Here, Network layer acts as a Control Layer for the hardware devices.

4. SDN SECURITY ISSUES AND SOLUTIONS

The traditional network offers a variety of advantages, some of which are discussed below. In Traditional networks, the network layer consists of Data and Control Plane. So, unlike SDN Traditional Network does not have separate Data and Control layer. The combined Data and Control layer has its advantages. Traditional Network supports higher bandwidth performance and low response time as compared to SDN [9]. Figure 2 shows the basic architecture of Traditional Networks which consists of two layers, Application Layer and Network Layer.

Another biggest advantage is the security. The Control layer defines security and forwarding policies and one defined device needs to be configured separately again to change the network policy. It becomes very difficult for a malicious user to re-configure each device. But in SDN, once Control Plane is compromised, the whole network will go down at once. In SDN single point of failure is present in the Control Layer. South-bound Interface is not present, and the controller cannot be tapped or fed with false messages in Traditional Networks. Communication between controller and data layer cannot be tapped easily due to the tight coupling of these layers. Single Point of failure is very difficult in Traditional Networks, unlike SDN. Individual routers, switches, and gateways need to be attacked separately in Traditional Networks, which is a very cumbersome task for the attacker. Lack of centralized control makes Traditional Networks more immune to network attacks.

Even with these myriads of advantages provided by Traditional Networks, it is a cumbersome operation to update, reconfigure and manage the devices due to the absence of a separate Control Layer. But in SDN it is very

easy to manage the devices. In SDN, the control plane is centralized. Individual devices are not needed to be updated individually. Just by sending a command to Control Plane, all the devices are updated at once. SDN centralization of control plane has provided numerous benefits but it has also introduced Single Point of Failure for SDN Controller.

Attacks on SDN controllers are one of the most preferred choices of network intruders. In Traditional Networks, intruders must target individual routers and switches to halt the network but SDN centralization of important features has made their task straightforward. Spoofing attacks can be done in the Data Plane to steal confidential data without the knowledge of the administrator.

In the next section, we will discuss attack vectors for SDN Application, Control, and Data layer, with their latest countermeasures proposed by different researchers. We will also discuss the need for decentralized security management aspects of SDN, to provide enhanced security to a variety of threats.

5. ATTACKS ON THE CONTROL PLANE

In this section, we will enlighten various types of attacks that are possible in SDN Application, Control, and Data Layer, with the solutions proposed by the research community over a period. The effectiveness of each method is also evaluated. Security remains the foremost important issue in the SDN Architecture and the same is discussed below section. SDN architecture has not dealt with security issues in-depth and the security of SDN has downgraded when compared with the Traditional Networks. Layer-by-layer attacks with their countermeasures are also discussed in depth.

A. DoS/DDoS Attacks on the Control Plane

Distributed Denial of Service (DoS) Attacks on SDN Controller obstructs legitimate user requests by exhausting the memory and CPU resources. Various types of DDoS attacks like SYN Flood, HTTP flood, Ping of Death, and low-rate DDoS attacks can cause a single point of failure in the SDN controller. These attacks with their countermeasures given by the research community are discussed comprehensively.

SDN suffers from a PACKET-IN bombarding attack. Network Intruder can generate a variety of malevolent packets from several IP addresses. Intruders can generate many BOT's by sending malicious code to them. These IP addresses are also spoofed so that identity of the host also cannot be checked and trusted. Once a packet arrives from an unknown IP address whose entry is not present in the OpenFlow Switch TCAM table, the packet is sent for inspection to the SDN controller. SDN Controller generates a new flow table rule and sends back the rule to the OpenFlow Switch. But if the intruder generates many unknown packets, OpenFlow Switch will pass a large number of PACKET-IN messages that will overwhelm the SDN controller. Legitimate user requests cannot be processed now.

Attackers can also manipulate TCP three-way handshake to generate TCP SYN Flood attacks. During this attack, the intruder bombards the switch with lots of TCP SYN messages. This creates a lot of half connections that quickly depletes the system resources and legitimate users cannot now make a connection with the switch. Dang et al. [10] proposed an SDN-based SYN Proxy Framework (SSP) to mitigate TCP SYN Flood attacks. Real-time traffic is analyzed of around 100 servers to get the typical time-out period in the case of normal traffic. They found out that in normal traffic TCP Handshake process finishes in less than 2 seconds. This time is calculated dynamically to delete half-open connection faster from the OpenFlow Switch to free switch resources. They developed an application named SYN Proxy Module (SPM) that runs in an SDN controller. For each flow entry, SPM calculates a time-out period. If the timeout exceeds the threshold value, the entry is removed from the OpenFlow Switch flow table removing the half-open TCP SYN connection. The advantage of this method is its easy deployment since it does not require any hardware or protocol revision or extension.

Mohammadi et al. [11] proposed SLICOTS, a lightweight TCP Handshake detection mechanism. First, temporary rules are inserted in the flow table, and only upon validation permanent rules are inserted. The proposed method takes additional memory when many users are present in the network. Fichera et al. [12], presented An OpenFlow-based REmedy (OPERETTA) to mitigate TCP SYNFLOOD Attacks against web servers that block the MAC Address that crosses the threshold. An attacker can surpass OPERETTA easily by just completing TCP Three-way Handshake Mechanism. Both OPERETTA and SLICOTS are unable to distinguish between the flash crowd and DDoS traffic. In other words, which is legitimate traffic, and which is malevolent traffic.

Ravi et al. [13] proposed AEGIS, a lightweight TCP SYN Flood detection mechanism. AEGIS application runs in the SDN Controller and uses two modules Assessment for detection and Analysis for prevention. In the Assessment phase, eight metrics are used to check whether degradation in service has happened or not in SDN Controller. These separate eight values are converted into a standalone value using Merrill's Figure of Merit (FOM). FOM is further divided into five stages ranging from low to high performance of the SDN Controller. If the FOM value lies in medium, low, or very low category then for further investigation stage 2 is deployed. In stage 2, the proposed algorithm checks whether it is flash crowd traffic or DDoS traffic. This method cannot perform early detection of DDoS attacks.Only when the performance of the SDN controller has degraded, the controller finds the attack.

Man-in-the-Middle attack intercepts communication be-

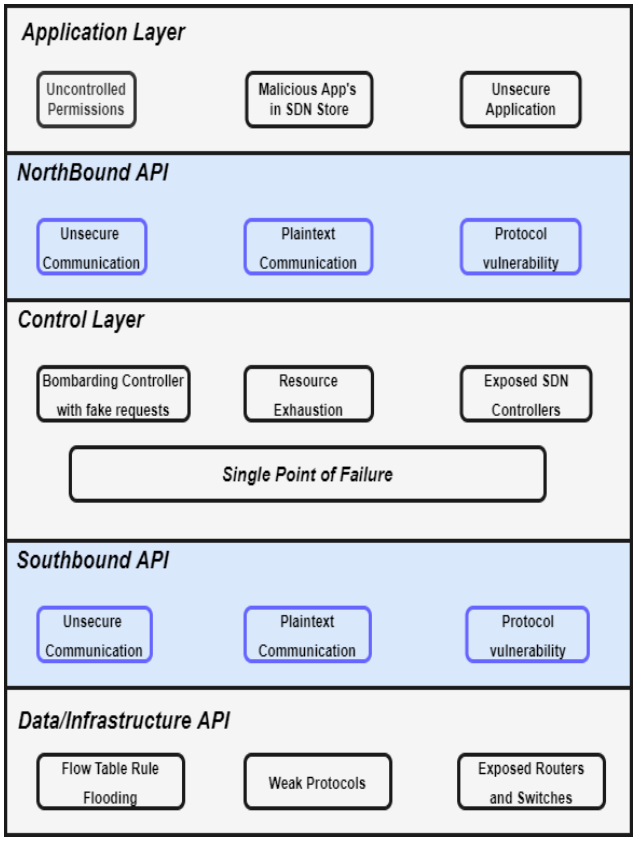

Figure 3. Vulnerabilities present in different layers of SDN.

tween Client and SDN controller. Once communication done by the SDN controller is intercepted, there will be no privacy of data in the whole network. Address Resolution Protocol (ARP) Spoofing is used to intercept communication between client and controller. The attacker forges the address table in the SDN controller and thus controller sends packets to the attacker. Brooks et al. [14] performed an ARP Spoofing attack on Open Daylight Controller using Kali Linux Operating System. Three physical machines were used in this experiment. One running ODL Controller, another running Kali Linux OS that catches communication between controller and host. The third machine runs SDN hosts using the ODL web interface.

Using the kali Linux ettercap tool, communication between controller and host passes first through Kali Linux. Kali Linux then intercepts the communication and was able to sniff the username and password of the ODL web interface. ODL Controller uses HTTP plaintext mechanism instead of secure HTTPS and thus attacker was able to sniff network credentials. This small vulnerability can compromise the full privacy of the network and MITM attacks are hard to detect. The author proposed the use of third-party tools to detect MITM attacks. Zhang et al. [15] proposed CMD as a proactive detection mechanism to detect MITM attacks like ARP, DNS, DHCP, ICMP spoofing. Bloom filters are used to detect MITM and hence no packet inspection is done to detect MITM attacks. CMD catches all the flow information in the network and it is checked whether any flow matches the flow description used in the MITM attack. CMD showed promising results like low positive rates and satisfactory performance.

In HTTP Flood Attack, the attacker bombards the SDN controller with legitimate GET or POST requests to exhaust its resources that will cause a denial of service to legitimate users. Sangeetha et al. [16] mitigated HTTP GET Flood attacks using SDN Controller. DDoS attacks are mitigated using traffic analysis by the webserver, but it consumes a significant number of resources on the webserver. Traffic from legitimate hosts is captured for some duration and their statistics is saved to be used later as a benchmark. If the traffic received is higher than the benchmark, a DDoS attack is in progress. Once a malicious host is detected, it is not blocked permanently but it is blocked for some duration to continue receiving data from the legitimate host. In the real world, multiple controllers are used, so this study has limited use in real-world networks.

A low-rate DDoS attack sends a burst of small DDoS traffic and it is very difficult to detect since its pattern is very similar to the genuine traffic. Large DDoS attacks can be detected easily but small and low-rate DDoS attacks can remain undetected for a long period of time. Zhijun et al. [17], proposed low-rate DDoS attack detection using Factorization Machine (FM). First, four features of flow rules namely duration of time flow rule are present in the Switch, several packets matched by the flow rule, Relative dispersion of match bytes (RDMB), and Relative dispersion of packet interval (RDPI) are extracted and are passed to FM Machine learning Algorithm for detection of DDoS attack. FM Machine Learning Algorithm combined with multi-feature combination achieved a detection rate of 95.8 percent. FM used a large amount of training data which is a time-consuming process and early detection of DDoS attacks is also not possible.

Zhu et al. [18], proposed a timeout mechanism to dynamically adjust the entry's time period in the flow table. The current timeout method available in the SDN is static, not dynamic which leads to the large number of PACKET-IN messages sent to the controller. The controller becomes overwhelmed and the resource consumption on the SDN controller increases. Load Aware feedback Algorithm optimizes the flow table limited resources and reduces the amount of PACKET-IN messages sent to the SDN controller. This mechanism can be used to stop overwhelming the SDN controller in case of a low-rate DDoS attack.

To detect DDoS attacks, researchers have developed their own customized algorithms. But these algorithms needs to be developed again and again over the period of time, to strengthen their performance. Instead of developing and upgrading customized solutions, solutions available

already in the market can be used. Badotra et. al [19] used SNORT Intrusion Detection System (IDS) to detect DDoS attack on ODL and ONOS Controller in its early phase. Once an attack is detected in its earlier phase, mitigation steps can be taken to thwart the attack before the entire network goes offline. New rules were written inside the SNORT rules directory to detect DDoS attacks. SNORT is one of the oldest and leading IDS system and its detection rate is undoubtedly remarkable. But SNORT uses a single processor to process the data. SDN networks process gigabits of data and SNORT performance to process that much data effectively, remains doubt full.

B. Countermeasures of DoS/DDoS Attacks

A myriad of customized algorithms is proposed by the research community to provide SDN security, some of which are discussed in the previous section. In this section, we discuss the benefits and limitations of the most common techniques used to secure SDN layers.

1) Entropy-Based Solutions:

The entropy-based approach can be used for earlier and lightweight detection of DDoS attacks. Mousavi et al. [20] gave the first idea of using entropy to detect earlier and timely detection of such attacks. This method detects attacks in the first stage, but it suffers from high false-positive rates. Different methods based on the variation of information entropy are proposed by the research community over a period. Li et al. [21] proposed a lightweight detection scheme using Information Entropy to detect DDoS attacks. In their approach, instead of collecting and analyzing the data of the flow table they have extracted and analyzed PACKET-IN messages. They observed in PACKET-IN message three fields namely, the entropy of fake IP source address increases and the entropy of IP destination address and port address decreases during DDoS attack. Detection mechanism schemes based on the Information Entropy approach cannot detect well-known DDoS attacks and choosing the threshold value is also a challenging task. Further only simulated results were shown and whether this approach will work in a real-world environment or not is unconfirmed. Sudar et al. [22] had combined entropy and machine learning approaches to reduce false positives in a DDoS attack. Early detection of DDoS attacks is performed by information entropy, and the false positives are reduced in the second stage by the deployment of machine learning algorithms.

In nearly all entropy-based solutions, if the attacker sends data from a spoofed IP Address to a random destination, the entropy approach will result in a high false-positive rate. Hybrid approaches that combine the entropy method with other methods are recently introduced by the research community. Shohani et al. [23] proposed a three-stage approach to detect such types of random DDoS attacks. Here, the trapezoidal relation between the number of received packets and the missed packets is analyzed. In stage 1, the controller asks OpenFlow switches for the total number of received packets and missed flow table entries. In stage 2, using Exponentially Weighted Moving Average (EMWA) method two thresholds are calculated to create the trapezoidal pattern. In stage 3, if the table switch misses entries exceeding the computed threshold then a DDoS attack is in progress. Figure 3 shows list of vulnerabilities present in all the three layers of SDN.

AbdelAzim et al. [24] proposed another two-stage hybrid approach. They combined entropy and the KL method to detect concurrent DDoS attacks that were not possible in other hybrid approaches. Their assumption follows that till attack network traffic follows a specific constant pattern. When an attack occurs, this normal traffic flow changes abruptly, and the comparison of these two-time internals will yield that an attack is in progress. Probability Density functions (PDF) are created using the count of source and destination IP addresses. Distance between PDFs at any two successive times internal is calculated using the KL method to check whether the normal behavior of the network has changed or not. The distance will rise during the starting phase and ending phase of the attack leading to the detection of a DDoS attack. Upon successful detection, malicious nodes are first banned and a first and final warning is given to them, failing which they will be blocked indefinitely.

Blockchain is a decentralized, distributed ledger that securely keeps a record of transactions. Blockchain is majorly used to secure digital crypto-currency transactions like Bitcoin [25]. Other applications of Blockchain are like securing medical research data, voting mechanisms, and cross-border payments data. Blockchain consists of Blocks that are connected in a chain, hence the name Block-Chain.

2) Blockchain-Based Solution to secure SDN

Before discussing Blockchain integration with SDN, we discuss briefly about the functionality of Blockchain. Blockchain consists of chain of blocks. Each block consists of a hash of data of the previous block, timestamp value, version information, a nonce, and Merkle tree root hash. Genesis Block is the parent block in the Blockchain, and it contains the private key. It is practically impossible to alter and delete transactions once they are added to Blockchain. If an attacker wants to alter a Block, he must change the previous block hash also. If the attacker wants to change the previous node, he must change the previous previous node also. In a way, the attacker must change all the nodes in a Blockchain simultaneously to avoid detection and that is nearly impossible. Blockchain enjoys heightened security architecture and is a proven technology to secure crypto-currency.

Each block contains a block header and block body. Block header contains a 256-bit hash of data of the previous block, timestamp value, version information, a nonce, and Merkle tree root hash that contains the hash of all transactions in a block. The body contains the transaction counter

and transaction. The size of the block decides how many transactions (TX) a block can handle. Popular Blockchain platforms are Ethereum Blockchain, Hyperledger, Ripple, Corda, Open ledger, IBM Blockchain, etc.

Using Smart Contracts in Blockchain provides variety of advantages. For the transaction management in Blockchain mining, which is performed by miners, requires some computational energy of the processing devices. This task is cumbersome and can be made easier by using smart contracts of Blockchain. Smart Contracts avoid the necessity of any Third Party involvement for the Proof of Work. Based on the protocols written inside a Smart Contract, transactions are added to the Blockchain. Ethereum Blockchain uses the Solidity programming language to create Smart Contracts. Blockchain provides myriad of benefits like enhanced security, distinguished security, data integrity, transparency and instantaneous traceability. This scenario is depicted in figure 5.

In a Multi SDN Controller environment, Blockchain can be added horizontally or vertically. As a horizontal layer, Blockchain sits between SDN controllers and the Data Infrastructure layer to secure communication between them. As a vertical layer, Blockchain is used to secure communication between SDN controllers itself to provide more security.

Following proposals are given by research community to secure SDN Controller using Blockchain. Basnet et al. [26] proposed BSS (Blockchain Security Over SDN) that uses Blockchain to securely transfer files between SDN nodes. Using mininet emulator, 10 hosts are created and Open Daylight is chosen as the SDN controller. To share a file securely between hosts Blockchain network is used. All 10 hosts are registered on the private Blockchain network. Serpent Programming is used to create smart contracts. Blockchain Pyethereum Platform issued to secure files by password using SHA256 encryption. Host h1 sends the file to host h5 by encrypting it with an h5 private key. So, only h5 can successfully decrypt the file. In this scenario, Blockchain ensures secure transmission of files in the SDN network.

Steichen et al. [27] proposed Chain Guard- A Firewall for Blockchain Applications. Some nodes in the SDN network are “Guarded Nodes”. These nodes are secured using Blockchain and they forward the traffic to other nodes in the Blockchain. Chainguard uses Open Daylight Controller. It can listen to the incoming Packet-in messages from the SDN Controller and can install new table flow rules to the connected OpenFlow switches. Chain Guard divides all nodes in the network into three categories namely Blacklisted, Whitelisted and Gray listed nodes. White-listed nodes can connect in the Blockchain network. Blacklisted nodes are the prohibited remote nodes and are not allowed to connect in the Blockchain. Gray-listed nodes are yet to be considered. This list contains IP Addresses, Ports, Transport protocol, timeout fields, etc. Once a Packet-in message is received by the OpenFlow Switch from the SDN controller, that entry is checked whether it is already present in any of the three lists. If not and if the entry does not violate the network rules, it is added to the gray list. Then, the request is passed to the next module and if the matching entry is found in the flow rule table or the whitelist, the packet is sent to the destination nodes. The entries in the Gray list are removed after the timer expires. Gray List capacity has been capped means it can receive only a limited number of requests. Once the gray list is full, new items cannot be added further. If the gray list becomes full, it shows an attack is in progress and all the entries in the gray list are dropped off. In this situation, some entries can still pass the gray list but most of them are dropped off. This architecture allows the working of SDN nodes even in case of a DDoS attack with reduced processing speed. SYN Flooding attack is done by the author and still, nodes were able to communicate with each other but with reduced speed. This architecture allows communication between Blockchain nodes with limited packet processing in case of a DoS or DDoS attack.

DoS/DDoS attacks like TCP SYN flooding attacks can be detected using packet sniffing tools. Once an attack is detected, the attacker’s IP Address is added to the Ethereum Blockchain. Now all nodes in the network know the address of the attacker and hence they will not receive any communication from them. The author created a DDoS attack using hping3 tool. Once the random IP packets are recognized, their IP addresses is added in a file and are shared with the Blockchain network. These IP addresses will be blacklisted; hence further communication will be suspended from these IP Addresses. This architecture allows efficient processing of packets by the OpenFlow Switches.

Single SDN controller suffers from Single Point of Failure and scalability issues. To prevent this type of failure, multi-SDN controller architecture is proposed. However, the multi-SDN controller suffers from inconsistency and scalability issues. Yazdinejad et al. [28] tried to solve these issues using Blockchain technology by combining both horizontal and vertical architecture of SDN controller. The Control layer is changed and now uses both horizontal and vertical architecture for controllers. The Control layer contains domains that contain various SDN controllers in vertical order. Blockchain is applied between horizontal SDN controllers to record network information that ensures data integrity and reliability. Table 1 shows comparison of different Blockchain-Based Security Mechanisms for SDN Controller.

Each table entry in the SDN controller contains source IP, destination IP, domain name, state, priority, and time to live field. Similar entries are present for each and every link also. Blockchain between horizontal SDN controllers ensures state consistency and protection from malicious attacks from the outside environment. Table 1 shows a com-

TABLE I. Blockchain Based Security Solutions for SDN Controller

Comparison of different Blockchain-Based Security Mechanisms for SDN Controller			
S.No	Methods used in SDN	Features of the Mechanism	Limitations
1.	BSS (Blockchain Security Over SDN)	Solves file transfer security issues in SDN. Securely transfer files between SDN Hosts using Blockchain Technology. Uses receiving host public key as the password.	File encryption algorithms, use of private and public key delays communication speed and this method should be optimized for large files.
2.	Chain Guard- A Firewall for Blockchain applications	Uses attacker IP address to block communication during DDoS attack. Attackers' IP addresses are stored in the Blockchain to permanently stop communication from illegitimate nodes.	In case of flooding attack from multiple sources, network works with reduced bandwidth only.
3.	Causing a DDOS attack and mitigating it using Blockchain and smart contracts	Enhances detection of DDoS attacks by using netstat and nmap to detect spoofed IP addresses. Blacklists IP address that causes DDoS attack and store them permanently using Blockchain.	Log files created are very large and should be stored efficiently. Nmap detects DDoS attacks but is slow. Packages other than nmap shall be deployed to gain processing speed.
4.	Block flow: A Decentralized SDN Controller using Blockchain	Uses Smart contracts to add or delete SDN switches. Removes security issues between SDN Control and data layer like DDoS, Single Point of failure of SDN Controller.	Devices are added manually to the network, and this design may suffer from "Ice Age Problem".
5.	BCFR: Blockchain-based Controller Against False Flow Rule Injection in SDN.	Communication between the Control layer and Data Layer is not secure and is susceptible to false flow injection rule attacks. The proposed solution uses Blockchain to secure against the attack.	Time taken by the Attack Prevention stage is not evaluated and its performance is unknown.
6.	Brain Chain - A Machine-learning Approach for protecting Blockchain applications using SDN	Protects Blockchain applications against DNS Amplification Attacks and has a low false positivity rate.	Machine learning algorithms need to be trained and hence cannot detect DDoS attacks in the early stage.
7.	A Blockchain-based security model for SDNs	Protects SDN Control and Data Plane using specialized algorithms to protect them against myriad attacks. The proposed algorithm takes O(n*logn) time which is linear and satisfactory.	Does not clearly define what type of attacks the proposed algorithm can protect from. Additional Blockchain blocks ABC, CBC, and DBC introduce additional overhead and delays communication speed.

parison of various Blockchain security techniques, proposed by the research community for SDN, with their features and limitations.

Theviyanthan et al. [29] proposed Block Flow, A Decentralized SDN Controller using Blockchain. Over-centralization of the SDN control layer has decreased its security by introducing a Single Point of Failure. Introducing Blockchain in the Control layer decentralizes its features. Ethereum is used as the Blockchain and Solidity is used as a programming language to develop a smart contract. Using Smart Contract switches can be added or deleted in the SDN network. Blockchain solves the Single Point of Failure of the SDN Controller, the DoS/DDoS attack, and the communication issue between the Control and Data layer. This method can also be deployed on multiple SDN controllers and offers several advantages like state synchronization across all devices, even distribution of workload, and permanent storage of log files for future security auditing.

Boukria et al. [30] presented a Blockchain-based controller to protect against false flow rule Injection attacks. Two separate stages, Attack Detection, and Attack Prevention are applied to secure communication between SDN Controller and OpenFlow Switches. In the attack detection phase, the VMFw node compares the rule sent by the attacker with the rules stored in the Blockchain. If both rules are not matched, then the attacker is blocked permanently. Performance of Attack Detection stage is evaluated and shows promising results. Stage 2, namely the attack prevention stage is not evaluated, and its performance remains unknown.

Houda et al. [31] proposed Brain chain, which uses a Machine learning algorithm with Blockchain for protecting SDN nodes against Domain Name System (DNS) amplification attacks. Brain chain uses an information collection scheme called FS (Flow statistics collection scheme) to collect features of flow, an entropy-based scheme called ES, a machine learning algorithm called BF (Bayes Network-based Filtering scheme) to detect network anomalies, and a DNS Mitigation (DM) scheme to remove illegitimate DNS requests. Using a combination of FS, ES, and BF DNS Amplification attack is detected and is removed using DM scheme.

Lokesh et al. [32] proposed three Blockchain Networks namely Application Blockchain (ABC), Device Blockchain (DBC), and Controller Blockchain (CBC). CBC block contains log files, Metadata, previous block hash, and root block hash. DBC contains Merkley Hash Values to validate transactions of devices in a cluster. ABC and CBC Blockchain prevent malicious activity at the Northbound API and CBC prevents SDN controller from Single Point of Failure.

C. Machine Learning Techniques to prevent DoS/DDoS

Various Machine learning algorithms have been proposed by the research community over the years to detect and prevent SDN layers. Artificial neural networks, Self-Organizing Maps, Bayesian Networks, and fuzzy logic are used to create an Intrusion Detection System (IDS). Here, we will discuss only dominant and latest findings in this area.

Braga et al. [33] proposed a lightweight technique that uses Self-Organizing maps to detect DDoS attacks on the NOX/OpenFlow controllers. Their algorithm used three modules flow collector, feature extractor, and classifier that classifies the traffic as benign or malicious based on six parameters. These include Average of Packets per flow (APf), Average of Bytes per flow (ABf), Average of Duration per flow (ADf), Percentage of Pair-flows (PPf), Growth of Single-flows (GSf), and Growth of Different Ports (GDP) that filters malicious traffic. This algorithm ensures very few false negatives and a very high rate of detection. This algorithm cannot work on multiple controllers, only perform detection not mitigation, and experiments are collection and analysis of data will overwhelm and degrade controller performance.

Dotcenko et al. [34] proposed an Intrusion Detection System (IDS) for SDN using fuzzy logic. These IDS used Rate Limiting Algorithm, TRW algorithm, and fuzzy logic. The combination of this security algorithm with fuzzy logic showed better results than the security algorithms used separately. Using this approach, fewer lines of code have to be written and external modules can be easily added to the existing code.

Chung et al. [35] proposed NICE, A Multiphase Distributed vulnerability detection, measurement, and countermeasure selection mechanism, that prevents vulnerable virtual machines to be compromised in the cloud. NICE creates an attack graph based on three parameters Cloud System Information, Virtual network topology and configuration information, and Vulnerability information that identifies atomic attacks which can lead a system to an undesirable state. NICE cannot work alone and needs additional host-based IDS to improve attack detection accuracy.

Dillon et al. [36] proposed a DDoS mitigation algorithm that utilizes the OpenFlow protocol used in SDN. OpenFlow Switch stores flow entries in the Ternary Content-Addressable Memory (TCAM) table. It consists of three phases Initial Detection, Identification, and Blocking phase. In Initial Detection Phase, Standard Deviation is calculated of packet/byte rate for the last 60 entries of the flow table. The difference between expected and real deviation is calculated to detect anomalies that can cause a DDoS attack. In the Identification phase, the traffic sent by the initial detection phase is again filtered to remove false negatives. Packet Symmetry is used to filter malicious packets and traffic is blocked temporarily to detect malicious clients

sending DDoS traffic. Finally, malicious traffic is blocked using their IP address. OpenFlow Switches have limited memory available for the TCAM tables which makes this application have limited performance and use in the real world.

Rahman et al. [37] evaluated state-of-the-art machine learning algorithms like J48, Random Forest (RF), Support Vector Machine (SVM), and K-Nearest Neighbors (K-NN) that are used to detect and block DDoS traffic. DDoS traffic is captured for 30 minutes and legitimate traffic for 3 hours. This data is preprocessed and is used to train and test the above four machine learning algorithms. J48 classifier performed better in terms of accuracy, sensitivity, kappa statistic, and training time, testing time, recall, and precision than Random Forest, SVM, and K-NN Algorithms to detect DDoS traffic.

Perez-Diaz et al. [38] proposed IDS that used six machine-learning algorithms to prevent low-rate DDoS attacks that are difficult to detect. Low-Rate DDoS attacks like DDoS Sim, GoldenEye, H.U.L.K., R.U.D.Y., Slow Body, Slow Headers, Slowloris, and Slow Read are performed on the ONOS Controller. HTTP flows are collected in the Identification Phase where Flow Management Module creates JSON Object that indicates which Machine Learning Model to use. After identification is complete, the Suspicious Attacker list is installed in the ONOS Controller. In incremental stages, the flow from malicious users is dropped to lower the false positives.

IDS and IPS are separate and can be deployed on separate hardware due to the requirement of heavy processing power. Machine learning Algorithm solves security loopholes present in the layers of SDN, but machine learning algorithm requires training on the datasets that make them slow and training requires resource-intensive processing power. Due to the need for training the algorithms, DDoS attacks cannot be detected in the early stage. Machine learning algorithms can be combined with other attack mitigation schemes like entropy, to decrease the false positivity rate and to improve attack detection accuracy.

6. ATTACKS ON DATA PLANE

Attacks on Data Plane are directed to steal information from the South-bound interface. Sebbar et al. [39] found that popular SDN Controllers like Open Daylight (ODL), Ryu, and Open Network Operating System (ONOS) are susceptible to data-stealing Man-in-the-Middle (MITM) attacks. Virtual switch, router, and hosts were set up using the Mininet emulator. Kali Linux was used to capture information using data sniffing tools like ettercap and Wireshark. Experiments revealed that the ODL interface uses unsecured HTTP ports like 8080 and 8181. Due to this small vulnerability information transferred between the control and data layer of SDN can be tapped.

Cao et al. [40] proposed a stealthy attack on the SDN data plane. LOFT (Low Rate Flow Table Attack) was done to overflow flow tables to perform a DoS attack. The attack was done in two stages. In stage 1, namely, the Probing Phase, probing packets are sent to the switches to calculate the timeout configuration of flow rules. Packets that do not have their entries in OpenFlow switches will be sent to the SDN controller and hence will take a longer time to respond. In stage 2, the attacking phase, the attacker sends only minimum packets to overflow the flow tables causing a DoS attack. According to the author himself, the LOFT attack is difficult to craft though not impossible.

SDN switches can be hijacked to initiate the attack on the SDN controller. Mitigating such attacks in a limited time frame becomes necessary to protect the network from further damage. Zhou et al. [41] deployed backup controllers that audit the network information to find compromised SDN controllers and switches. Network Update state information like addition, updating, and deletion of flow table entries are collected from both primary SDN controller and SDN switches by the auditor-controller. An audit ID is assigned to every network updating request. The auditor-controller re-executes every received request and adds its execution result to its audit record. If both records match, the request is from a legitimate user else the network device or controller has been compromised. Auditing requires extra storage space and takes a long time in processing and match audit records. It may happen that all the network switches may have been compromised before the auditor SDN controller detects the compromised devices. The proposed method also does not detect Saturation attacks targeted on OpenFlow switches.

Malicious Applications running on the Controller can generate new flow rules that can cause conflict with the existing rules proposed by the SDN controller in the SDN data plane. SDN allows large space i.e. 35 fields to specify rules from the users. So, it becomes difficult to enforce the rules defined by the users. Porras et al. [42] presented FortNox, a software extension for the NOX SDN controller to prevent conflicting flow rules. FortNox detects malicious flow rules and does priority-based filtering of flow rules. The priority is assigned to human administrators creating flow rules, second to security applications, and third to other non-security applications. Using the Alias Rule Reduction Algorithm conflicting rules are detected and are removed according to their authorization roles.

Hu et al. [43] proposed Flow Guard, a robust firewall for SDN. Flow Guard does real-time detection using different modules like flow path space analysis, firewall authorization space, and packet blocking. Experiments were done in a real-world environment and shows acceptable performance overhead.

Mai et al. [44] proposed Net Plumber, a real-time protocol-independent policy checking tool that uses Header Space Analysis (HSA) and dependency graph to check state

TABLE II. Security Solutions for Data Layer

Comparison of different security Mechanisms for the Infrastructure/Data Layer				
S.No	Attack Type	Proposed Solution	Features	Limitations
1.	Attacking SDN Switches	Backup Controllers	Backup controllers audit every transaction performed by the main SDN controller and if both results do not match, the attack is in progress.	The auditing process is slow and may report an attack when an attack has already infiltrated the system.
2.	Flow Rule Conflict in Data Plane	FortNoX	Using custom Alias Rule Reduction Algorithm, conflicting rules are detected and removed according to their authorization roles.	Works on old NOX Controller, has not been extended further and this prototype version has multiple loopholes present in it.
3.	Flow Rule Conflict in Data Plane	FlowGuard	Does real-time detection flow path space analysis and packet blocking.	Flowguard was integrated with Floodlight Controller, which is obsolete now. It has multiple performance issues, when deployed on real-world environment.
4.	Flow Rule Conflict in Data Plane	NetPlumber	Real-time, Protocol Independent Policy checking tool that uses Header Space Analysis and dependency graph.	Memory Intensive and a single computer system cannot process the same.
5.	Flow Rule Conflict in Data Plane	VeriFlow	Extra layer between SDN Controller and Data Plane that detects flow rule violation in real-time.	Has issues while working in a Multi-Controller Environment.
6.	Flow Rule Conflict in Data Plane	Anteater	Uses SAT to detect conflicting policy rules.	Has very low processing time making it unsuitable in a real-world environment.
7.	Flow Rule Conflict in Data Plane	ndb	A network debugger tool similar to a popular gdb debugger. ndb provides various debugging operations like backtrace, breakpoint, continue, etc	Only troubleshoots the flow rule issue but cannot trace the malicious application IP address so that attacker can be stopped from further attacking the network.
8.	Flow Rule Conflict in Data Plane	Brew	Works efficiently in multi-tenant SDN-based cloud environments where multiple applications make it difficult to detect flow rule conflicts.	The topology of the environment is not taken into account and thus performance and accuracy of the framework may suffer.

changes in real-time. Net Plumber is memory intensive and will not run on a single system. Multiple high-grade processors are required which will increase the infrastructure cost.

Haohui et al. [45] proposed Anteater, A Protocol Independent Tool that uses Boolean Satisfiability Problems (SAT) to detect conflicting policy rules. Anteater suffers from low processing time and is not the best choice for real-world data centers. Khurshid et al. [46] proposed VeriFlow which is an additional layer between SDN Control and Data Plane and detects flow rule violation in real-time. VeriFlow divides the flow rule into equivalence classes and checks network configuration in real-time to find buggy and malicious applications causing disrupting flow rules. VeriFlow's performance seems remarkable since it can do the verification of rules in hundreds of microseconds.

Handigol et al. [47] proposed ndb, which is a network debugger like gdb network debugger. NDB provides various debugging operations like back trace, breakpoint, continue, etc. Buggy or malicious OpenFlow Switches are recognized by ndb using ndb postcards. ndb postcards are sent by the OpenFlow switch to the collector for analysis and contain information like switch id, version id, and port number. All the techniques discussed above cannot detect cross-layer and multiple-layer flow rule conflict. Pisharody [48] proposed Brew-A Security Policy Analysis Framework that efficiently scrubs the flow table present in the Data layer and mitigates intrusion from buggy or malicious applications. The main characteristic of Brew is that it works in Multi-Tenant SDN-based Cloud Environment where multiple applications interacting with each other make it extremely difficult to manage policy conflicts.

7. ATTACKS ON APPLICATION PLANE

Northbound Interface provides communication between SDN application and controller. This interface provides two types of services. One is reading network state and the other is writing to the network state. Applications transfers an HTTP GET message to the controller and the controller interprets and sends the request to the data layer as an OpenFlow request. The data layer sends the relevant response to the controller and the controller responds with the result as an HTTP response to the application. Similarly, an application can send an HTTP POST request to the controller to install flow rules in the flow table. The controller then sends either success or failure to the application. This blind trust approach on Third-party applications causes several security vulnerabilities. Any application, whether legitimate or malicious can read and write to the network state since the origin of the HTTP requests is not considered into account. Behavior inspection check is not required for applications and hence legitimate applications can be turned into malicious ones without alerting the controller.

Lee et al. [49] showed how using small and lightweight malicious applications, three leading SDN Controllers namely Open Daylight, ONOS and Floodlight can be compromised. A malicious application is first uploaded on the SDN store. The once-installed malicious application stays undetected since SDN Controllers does not adopt a deep inspection mechanism for the newly installed third-party applications. Using this vulnerability, the author successfully tapped the PACKET-IN message in the case of the Floodlight controller. Their malicious application becomes the first one to receive a PACKET-IN message that can be used for malicious purposes. In ONOS Controller PACKET-OUT-ONLY parameter is manipulated by the malicious third-party application to downgrade the performance of the controller. In the Open Daylight controller, ArpHandler is a default application that manages ARP requests and replies. This application is attacked and deactivated by the malicious application so that the controller loses its functionality. A variety of solutions is proposed by the research community to prevent abusive third-party SDN applications.

Applications can ask SDN Controller for a complete view of the SDN Network. This information can be used for malicious purposes. So, an SDN application should be given only limited permissions as per their use. Granting all permissions to an application poses a security risk. For example, an SDN application that creates a circuit in the network should not be allowed to receive a PACKET-IN message. Network traffic information can be inferred from the same and can be used for further attacks. Granting complete control of the network state to the application is a security risk and should be mitigated.

Scott et al. [50] secured the Northbound Interface of SDN using Operation Checkpoint so that the SDN controller remains available to trusted applications only. Their model works like Android System in which an application is given limited permissions according to its usability. A simple gaming application should not be allowed to read contacts details in the Android system. Similarly, an Application should be given only limited permissions as per their usability. Logfile of the unauthorized operations to read and write the network state is also maintained. Intrusion Detection System can use this log file to block the malicious applications to prevent the system from further attacks.

Wen et al. [51] proposed SDN Shield, a lightweight mechanism that helps network administrators to enforce only essential permissions to the SDN applications. A variety of attacks can be done by compromising an application in the control plane. The author divided the attacks into four classes for which above mentioned approaches do not provide any protection. In Class 1 attack, PACKET-IN and PACKET-OUT messages can be used to manipulate the traffic between controller and data plane OpenFlow switches. In a Class 2 attack, a compromised application can attack a security application running on an SDN controller. In Class 3 attacks, Man-in-the-Middle and Blackhole attacks can be done by modifying OpenFlow flow table rules and in Class 4 attacks, sensitive information about the network can be

TABLE III. Security Solutions for Application Layer

This Table shows different attacks and proposed solutions for the Application Layer				
S.No	Attack Type	Proposed Solution	Features	Limitations
1.	Illegal use of Permissions by Third Party Applications	Operation Checkpoint	Similar to Android Permission System, where permissions are given to the applications only according to their use.	Covers only a limited amount of Permission attacks and requires manual processing by the administrator.
2.	Illegal use of Permissions by Third Party Applications	SDN Shield	Similar to the Android Permission System Model and covers four classes of permission attacks.	The process is not automatic but is manual, where network administrator is required for cross-checking, updating the permissions, API usage, etc.
3.	Illegal use of Permissions by Third Party Applications	ASTRAEA	Uses Natural language Processing (NLP) to check permissions. The process is nearly automatic, and the manual intervention of the administrator is minimal.	Implementation is tightly coupled with ONOS controller only. The same model cannot be used with other controllers like Open Daylight, floodlight, POX, etc.
4.	Illegal use of Permissions by Third Party Applications	VOGUE	Cross-Platform and will work on all leading SDN Controllers like Open Daylight, ONOS, Floodlight, etc.	SDN API should be well documented only then it will work effectively, and performance is also not up to the mark.
5.	Illegal use of Permissions by Third Party Applications	Verificare	Based on Verificare Modeling Language (VML) which verifies application properties like QoS, secuirty, reliability.	Does not perform Packet History and Packet Injection, that are needed for advanced debugging.
6.	DoS/DDoS on Application Layer	DDoS Detection and Mitigation using Deep Learning	Deep Learning is used to detect and mitigate HTTP-based fast and slow rate DDoS attacks.	Does not detect the attack in an early phase.

leaked by compromised applications using a side channel. SDN Shield protects against these four types of attacks. In SDN Shield administrator explicitly provides permissions to the applications. Whenever an application demands permission that is not assigned to it, the administrator is alerted.

This permission model is not completely fool-proof and suffers from a Permission and Semantic gap. SDN is a new technology and it is not easy for the network administrator to comprehend all the available features of a Third-party application. Hence, network administrators can allow some permission without their sufficient knowledge that can cause a security risk. Adversaries can trick end users to install malicious applications by giving a false description of the SDN application. Since, majority of end-users normally cannot understand the application behavior by looking at the description, they can be easily tricked by the adversaries to allow unnecessary permissions. The permission model of SDN is based on Android Permission Model that also suffers from the same issue. Manually processing permissions is not a satisfactory method and should be converted into an automatic process.

Kang et al. [52] resolved the permission model issues using an automatic tool called ASTRAEA, that runs on an ONOS controller. ASTRAEA uses static analytic tools using Natural Language Processing (NLP) to extract the necessary permissions and checks the same with the permission demanded by the application. ASTRAEA alerts the administrator in case an application demands more permissions than needed. ASTRAEA was further developed in the real world and positive feedback came from real world SDN developers. The downside of ASTRAEA is that its implementation is tightly coupled only for the ONOS controller.

Kang et al. [53] resolved this issue using VOGUE, which can be applied to any SDN Controller. VOGUE uses both manual and automatic methods for granting permission. VOGUE inputs SDN API documents and uses Natural Language Processing techniques to create a tree diagram that specifies which critical assets should be protected. After that SDN asset map is created that helps the network administrator to easily understand which application is accessing a specific resource and permission. VOGUE also provides generic and independent SDN Controller implementation and can be used with leading SDN Controllers like Open Daylight, Floodlight, ONOS, or RYU. VOGUE has some limitations like if SDN API is not well documented then its functionality will be limited. Performance is also an issue and should be optimized. According to the author, VOGUE is still in the prototype phase and the development is in progress.

DDoS or DoS attacks can be targeted against the application layer. Benzaid et al. [54] proposed a DDoS self-protection framework based on Deep Learning and SDN enablers. It consists of four modules namely Network Flow Collector, Features Extractor, Detector, and Security Policy Manager. It successfully detects and mitigates HTTP-based fast and slow rate flooding attacks on the SDN application layer.

SDN Architecture uses OpenFlow Random Host Mutation and Resonance for security purposes but they cannot defend against DDoS attacks. Jantila et al. [55] proposed a hybrid approach that uses Entropy and trust value to tackle DDoS attacks in the Application Layer of SDN. Based on Client behavior, trust values are assigned to the users. Malicious users will have low trust value and legitimate ones will have a higher one. Even if a malicious user gains high trust value to misbehave later, he is caught using the information entropy approach. Using two filters makes malicious users difficult to infiltrate the system. Finally, the vulnerability of SDN is checked using the STRIDE threat model [56].

8. MISSING LINKS IN SDN SECURITY

1) Less focus on enhancement of SDN Controllers Security

Open-Source SDN Controllers like ODL, ONOS, RYU, POX, etc. process gigabits of network data per second. These Open-Source SDN Controllers are used as a backbone for the creation of customized SDN Controllers by commercial firms. These customized SDN Controllers are used to process live traffic, where the high performance of the servers becomes one of the top-most requirements for commercial firms. Due to the same, the SDN community has focused mainly on improving the performance of these controllers only. As these controllers are used in commercial markets, high performance becomes one of the most important factors in deciding their market share and popularity. For example, ONOS Controller depicts the best performance overall other SDN Controllers available in the market [57]. Hence, it has the largest market share so far.

Strengthening the security of SDN Controllers will somehow decrease the performance of SDN Controllers, as network data needs to be filtered out before use by the security algorithms. Due to this reason, security has been overlooked most of the time by the SDN community. But for a long-term benefit, security needs to be considered as the second-most important factor by the research community for SDN Controllers. As Single Point of Failure in the Control layer will make the entire SDN network useless for the customers. Hence, both performance and security both should be strengthened for SDN Controllers, before deploying them in a live environment.

2) Handful of Security Assessment Frameworks, that can automatically identify vulnerabilities present in the three layers of SDN

Researchers have developed a security assessment framework called DELTA, the first framework of its kind, that evaluates security performance of SDN environment in

depth [58]. DELTA provides systematic coverage of attacks that can be done on an SDN environment which was missing in earlier studies. With DELTA, network administrator can now find the loopholes and backdoors in the specific SDN environment, in a very limited time. Appropriate security countermeasures can now be applied to the SDN environment by the network administrator. Unknown attacks are also discovered using specialized fuzzy modeling tools. But still DELTA covers only a limited number of attacks, that be initiated by the nefarious users in SDN. New attacks like Race condition attacks, flow-based attacks and many similar attacks recently found by research communities, remains uncovered by DELTA [59] [60].

In case a new attack is identified by DELTA, manual intervention of network administrator is required to assess the attack using log files. The process of discovering vulnerabilities by DELTA is not fully automatic, and in manual intervention some process can be overlooked by the network administrator. Penetration testing frameworks that can automatically identify known and unknown vulnerabilities like DELTA, are very few in number. According to our best knowledge, DELTA is the only quality security vulnerability assessment framework that can identify in depth, known and unknown attacks on SDN layers.

3) Centralization of Control Layer without strengthening its Security

Centralization without strengthening security aspects causes Single-Point-of-Failure in SDN. In the case of a Traditional Network, every device has a firewall installed in it, and malicious users must penetrate all firewalls to enter the system. Even after penetrating the system, it is difficult to halt the entire network, due to the separation of individual hardware devices. In SDN, the control layer is the brain and does all processing for the devices. Once security mechanisms like firewall and IPS are bypassed, the entire SDN network will go down at once [61]. Due to the centralization of the control layer without strengthening its security, the security of the whole SDN network gets compromised in SDN.

Hence, techniques like Entropy, Machine-learning, Blockchain should be used individually or collaboratively to reduce single-point-of-failure presence in the Control layer.

4) Failure of In-Built Security Mechanisms of SDN Controllers

ODL and ONOS are the top two leading Open-Source SDN Controllers available in the market [62]. Other Commercial SDN solutions also use either ODL or ONOS as the backbone to create their customized SDN software.

DDoS attacks on Control Layer are the most dreadful attacks, as they will cause Single-Point-of-failure, which in turn will make the entire SDN network offline. So, to thwart these attacks ODL community has created the Defense4All application, which is a dedicated DDoS detection and mitigation application [63]. Similarly, ONOS uses Security Mode ONOS application to thwart web-based attacks. Both Defense4All and Security Mode ONOS fails to protect against DDoS attacks. Defense4All uses traffic redirection to divert DDoS traffic. This approach fails to protect SDN Controllers when multiple nodes are used by the attacker to send network traffic [64]. Similarly, other leading SDN controllers like ONOS, RYU, Beacon, Floodlight, etc. are also prone to web-based attacks. Either SDN developers should deploy third-party DDoS mitigation software or shall enhance the existing in-built mechanism to mitigate these attacks. As in Control Layer Single Point of Failure is present, DDoS attack on Control Layer would make the entire SDN network offline. Hence, this security gap should be fixed by ODL and ONOS community.

9. CONCLUSION

A famous proverb says, “A Chain is strong as the weakest link” and in SDN, security is the weakest link. Similarly, centralization of the control layer, without strengthening its security, has degraded SDN security exponentially. Centralization is a major advantage of SDN over Traditional networks, but without enhanced security, it creates a Single Point of Failure. This Single-Point-of-Failure makes the entire SDN Chain defenseless against even naive DDoS attacks. So, centralization of the control layer has become a disadvantage itself due to lack of security.

Other layers of SDN, like Application and Data Layer, are also not immune to web-based attacks. Multiple vulnerabilities have been found in these layers of SDN. In the Application layer, illegal use of permissions by Third-Party applications jeopardizes the security of the SDN network. In the Data layer, attacks like MITM are used to steal information flowing between Control and Data Layer.

To provide security for SDN layers many techniques like Entropy, Machine Learning, Blockchain, etc. can be deployed. One such new technique that seems to be promising is Blockchain, which is widely used over the internet to secure cryptocurrency like Bitcoin. Blockchain can be integrated with SDN to provide decentralized security benefits to the SDN. So far, not much research has been done to successfully and completely integrate these two different technologies. Blockchain provides trustworthy security benefits that Conventional Security Algorithms cannot provide. Other techniques like Entropy and Machine Learning are also in the development phase to protect SDN Layers.

Security assessment frameworks like DELTA that can automatically detect vulnerabilities present in SDN are very limited in number. Similar cross-penetration penetration testing tools are necessary to automatically assess the degree of security of SDN. SDN security is a road far less traveled. SDN is still in its juvenile phase of development, particularly in the case of security. SDN provides a myriad of benefits over Traditional Networks, but its security should be furnished deeply and systematically by the research

community, to get its full benefits.

References

[1] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[2] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (sdn): a survey," *Security and communication networks*, vol. 9, no. 18, pp. 5803–5833, 2016.

[3] "Software-defined networking (sdn) definition," accessed: 2022-03-14. [Online]. Available: https://opennetworking.org/sdn-definition/

[4] "Architecture and internals guide," accessed: 2022-03-14. [Online]. Available: https://wiki.onosproject.org/display/ONOS/Architecture+and+Internals+Guide

[5] N. Bizanis and F. A. Kuipers, "Sdn and virtualization solutions for the internet of things: A survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

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

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

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

[9] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[10] V. T. Dang, T. T. Huong, N. H. Thanh, P. N. Nam, N. N. Thanh, and A. Marshall, "Sdn-based syn proxy—a solution to enhance performance of attack mitigation under tcp syn flood," *The Computer Journal*, vol. 62, no. 4, pp. 518–534, 2019.

[11] R. Mohammadi, R. Javidan, and M. Conti, "Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 487–497, 2017.

[12] K. Kemp, J. Griffiths, S. Campbell, and K. Lovell, "An exploration of the follow-up up needs of patients with inflammatory bowel disease," *Journal of Crohn's and Colitis*, vol. 7, no. 9, pp. e386–e395, 2013.

[13] N. Ravi, S. M. Shalinie, C. Lal, and M. Conti, "Aegis: Detection and mitigation of tcp syn flood on sdn controller," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 745–759, 2020.

[14] M. Brooks and 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*, 2015, pp. 45–49.

[15] K. Zhang and X. Qiu, "Cmd: A convincing mechanism for mitm detection in sdn," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2018, pp. 1–6.

[16] R. Sanjeetha, K. A. Shastry, H. Chetan, and A. Kanavalli, "Mitigating http get flood ddos attack using an sdn controller," in *2020 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*. IEEE, 2020, pp. 6–10.

[17] W. Zhijun, X. Qing, W. Jingjie, Y. Meng, and L. Liang, "Low-rate ddos attack detection based on factorization machine in software defined network," *IEEE Access*, vol. 8, pp. 17 404–17 418, 2020.

[18] H. Zhu, H. Fan, X. Luo, and Y. Jin, "Intelligent timeout master: Dynamic timeout for sdn-based data centers," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 734–737.

[19] S. Badotra and S. N. Panda, "SNORT based early DDoS detection system using opendaylight and open networking operating system in software defined networking," *Cluster Computing*, vol. 24, no. 1, pp. 501–513, May 2020. [Online]. Available: https://doi.org/10.1007/s10586-020-03133-y

[20] S. M. Mousavi and M. St-Hilaire, "Early detection of ddos attacks against sdn controllers," in *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015, pp. 77–81.

[21] R. Li and B. Wu, "Early detection of ddos based on phi-entropy in sdn networks," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1. IEEE, 2020, pp. 731–735.

[22] K. Muthamil Sudar and P. Deepalakshmi, "A two level security mechanism to detect a ddos flooding attack in software-defined networks using entropy-based and c4. 5 technique," *Journal of High Speed Networks*, vol. 26, no. 1, pp. 55–76, 2020.

[23] R. B. Shohani and S. A. Mostafavi, "Introducing a new linear regression based method for early ddos attack detection in sdn," in *2020 6th International Conference on Web Research (ICWR)*. IEEE, 2020, pp. 126–132.

[24] N. M. AbdelAzim, S. F. Fahmy, M. A. Sobh, and A. M. B. Eldin, "A hybrid entropy-based dos attacks detection system for software defined networks (sdn): A proposed trust mechanism," *Egyptian Informatics Journal*, vol. 22, no. 1, pp. 85–90, 2021.

[25] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE international congress on big data (BigData congress)*. IEEE, 2017, pp. 557–564.

[26] S. R. Basnet and S. Shakya, "Bss: Blockchain security over software defined network," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2017, pp. 720–725.

[27] M. Steichen, S. Hommes, and R. State, "Chainguard—a firewall for blockchain applications using sdn with openflow," in *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*. IEEE, 2017, pp. 1–8.

[28] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, Q. Zhang, and K.-K. R. Choo, "An energy-efficient sdn controller architecture for iot

networks with blockchain-based security," *IEEE Transactions on Services Computing*, vol. 13, no. 4, pp. 625–638, 2020.

[29] T. Krishnamohan, "Blockflow: A decentralized sdn controller using block-chain," *Theviyanthan Krishnamohan, Kugathasan Janarthanan, Peramune PRLC, Ranaweera AT (2020)*, 2020.

[30] S. Boukria, M. Guerroumi, and I. Romdhani, "Bcfr: Blockchain-based controller against false flow rule injection in sdn," in *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1034–1039.

[31] Z. Abou El Houda, A. Hafid, and L. Khoukhi, "Brainchain-a machine learning approach for protecting blockchain applications using sdn," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[32] B. Lokesh and N. Rajagopalan, "A blockchain-based security model for sdns," in *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE, 2020, pp. 1–6.

[33] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *IEEE Local Computer Network Conference*. IEEE, 2010, pp. 408–415.

[34] S. Dotcenko, A. Vladyko, and I. Letenko, "A fuzzy logic-based information security management for software-defined networks," in *16th International Conference on Advanced Communication Technology*. IEEE, 2014, pp. 167–171.

[35] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "Nice: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE transactions on dependable and secure computing*, vol. 10, no. 4, pp. 198–211, 2013.

[36] N. Z. Bawany, J. A. Shamsi, and K. Salah, "Ddos attack detection and mitigation using sdn: methods, practices, and solutions," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425–441, 2017.

[37] O. Rahman, M. A. G. Quraishi, and C.-H. Lung, "Ddos attacks detection and mitigation in sdn using machine learning," in *2019 IEEE World Congress on Services (SERVICES)*, vol. 2642. IEEE, 2019, pp. 184–189.

[38] Arturo, I. A. Valdovinos, K.-K. R. Choo, and D. Zhu, "A flexible sdn-based architecture for identifying and mitigating low-rate ddos attacks using machine learning," *IEEE Access*, vol. 8, pp. 155 859–155 872, 2020.

[39] A. Sebbar, M. Boulmalf, M. D. E.-C. El Kettani, and Y. Baddi, "Detection mitm attack in multi-sdn controller," in *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*. IEEE, 2018, pp. 583–587.

[40] J. Cao, M. Xu, Q. Li, K. Sun, Y. Yang, and J. Zheng, "Disrupting sdn via the data plane: a low-rate flow table overflow attack," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2017, pp. 356–376.

[41] H. Zhou, C. Wu, C. Yang, P. Wang, Q. Yang, Z. Lu, and Q. Cheng, "Sdn-rdcd: A real-time and reliable method for detecting compromised sdn devices," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2048–2061, 2018.

[42] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for openflow networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 121–126.

[43] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "Flowguard: building robust firewalls for software-defined networks," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 97–102.

[44] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King, "Debugging the data plane with anteater," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 290–301, 2011.

[45] H. Mai, A. Khurshid, R. Agarwal, and Caesar, "Debugging the data plane with anteater," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 290–301, 2011.

[46] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," pp. 15–27, 2013.

[47] N. Handigol, B. Heller, V. Jeyakumar, D. Maziéres, and N. McKeown, "Where is the debugger for my software-defined network?" pp. 55–60, 2012.

[48] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan, and D. Huang, "Brew: A security policy analysis framework for distributed sdn-based cloud environments," *IEEE transactions on dependable and secure computing*, vol. 16, no. 6, pp. 1011–1025, 2017.

[49] S. Lee, C. Yoon, and S. Shin, "The smaller, the shrewder: A simple malicious application can kill an entire sdn environment," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2016, pp. 23–28.

[50] S. Scott-Hayward, C. Kane, and S. Sezer, "Operationcheckpoint: Sdn application control," in *2014 IEEE 22nd International Conference on Network Protocols*. IEEE, 2014, pp. 618–623.

[51] X. Wen, B. Yang, Y. Chen, C. Hu, Y. Wang, B. Liu, and X. Chen, "Sdnshield: Reconciliating configurable application permissions for sdn app markets," in *2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2016, pp. 121–132.

[52] H. Kang, C. Yoon, and S. Shin, "Astraea: Towards an effective and usable application permission system for sdn," *Computer Networks*, vol. 155, pp. 1–14, 2019.

[53] H. Kang, V. Yegneswaran, S. Ghosh, P. Porras, and S. Shin, "Automated permission model generation for securing sdn control-plane," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1668–1682, 2019.

[54] C. Benzaïd, M. Boukhalfa, and T. Taleb, "Robust self-protection against application-layer (d) dos attacks in sdn environment," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.

[55] S. Jantila and Chaipah, "A security analysis of a hybrid mechanism to defend ddos attacks in sdn," *Procedia Computer Science*, vol. 86, pp. 437–440, 2016.

[56] S. Jantila and K. Chaipah, "A security analysis of a hybrid mech-

anism to defend ddos attacks in sdn," *Procedia Computer Science*, vol. 86, pp. 437–440, 2016.

[57] "Comparison of software defined controlling controllers," https://aptira.com/comparison-of-software-defined-networking-sdn- controllers- part-7-comparison-and-product-rating/, accessed: 2022-03-14.

[58] Z. Durumeric, Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J. A. Halderman, and V. Paxson, "The security impact of https interception." in *NDSS*, 2017.

[59] L. Xu, J. Huang, S. Hong, J. Zhang, and G. Gu, "Attacking the brain: Races in the sdn control plane," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 451–468.

[60] M. P. Singh and A. Bhandari, "New-flow based ddos attacks in sdn: Taxonomy, rationales, and research challenges," *Computer Communications*, vol. 154, pp. 509–527, 2020.

[61] "Benefits and the security risk of sdn: Isaca journal," Jul 2016. [Online]. Available: https://www.isaca.org/resources/isaca-journal/issues/2016/volume-4/benefits-and-the-security-risk-of-software-defined-networking#:~:text=Unlike%20traditional%20network%20design%2C%20SDN,well%2Ddefined%20application%20programming%20interface

[62] "Sdn controllers - a comparative approach to market trends,," Feb 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03133692/document

[63] Defense4all overview. https://www.radware.com/newsevents/pressreleases/ radware-releases-defense4all-industry-first-open-sdn-security-application-for-opendaylight-project. Accessed: 2022-03-14.

[64] H. Polat and O. Polat, "The effects of dos attacks on odl and pox sdn controllers," in *2017 8th International Conference on Information Technology (ICIT)*, 2017, pp. 554–558.

Mr. Pulkit Ohri is currently working as Senior Analyst (Digital Vertical) in eClerx, Chandigarh, India. He worked as Training Specialist in Department of Computer Science, Amity University, Madhya Pradesh, India. He completed Bachelor's (2014) and Master's Degree (2016) in Computer Science from University of Delhi. Currently he is pursuing PhD (Engineering), Computer Science from Amity University, Madhya Pradesh, India. His research area is Software Defined Networking Security enhancement.

Dr. Subhrendu Guha Neogi is currently working as Associate Professor in the Department of Computer Science and Engineering, Amity University Madhya Pradesh. He has been awarded Masters in Engineering, from West Bengal University of Technology. He has more than 20 years of experience in teaching. He served many reputed educational Institutes in India in various academic positions.