A survey on security measures on SDN

Prateekshya Priyadarshini
Department of Computer Science and Engineering
Indian Institute of Technology
Guwahati, Assam, India
ppriyadarshini@iitg.ac.in

Vijay Purohit
Department of Computer Science and Engineering
Indian Institute of Technology
Guwahati, Assam, India
vijay.purohit@iitg.ac.in

***Abstract*—Software-Defined Networks (SDN) is a promising area in Computer Networks in recent years as it introduces programmability to the Network leading to efficient Network Management. In SDN, Network Intelligence is centralized by separating data plane and control plane. Hence, controllers are highly vulnerable to various kinds of attacks which can destroy the entire network. In recent years, significant research is being done on the security measures to be taken for the possible attacks. In this paper, we present a survey on the types of measures being taken for safety of SDN, specifically the control plane. We present the background of SDN, typical attacks possible on SDN, various security measures taken for the safety of SDN, conclusion and future directions.**

***Index Terms*—SDN, Security, Network Programmability, Attacks, Measures, Blockchain, Cloud, IoT, ML.**

I. Introduction

Software-Defined Networks (SDN) is a network technology that came from the idea of programmable networks. SDN separates the data plane and control plane logically and provides programmability, reliability and flexibility to the network in order to improve network performance and monitoring. It centralizes network intelligence in one network, migrating it from traditional network management towards cloud computing. In [1] we can see that SDN mostly has three planes/layers i.e. control layer, infrastructure or data layer and application layer.

Control layer consists of the control logic that manages the forwarding table for each request of the switch based on the header of the packet. A network can have multiple controllers simultaneously that can manage a group of switches. However, it might get more hectic to handle the control plane. So, a central controller is selected from the group of controllers, and the others would be the backup controllers. In case of the central controller failure, we use the other controllers, which maintain the property of fault tolerance. SDN control plane can follow a centralized, hierarchical, or decentralized design. In the case of an entirely distributed control system where we have multiple controllers working simultaneously. There will be a central controller managing all the controllers. In this case, all the controllers will not be working as backup controllers.

Infrastructure layer consists of the switches which communicate with the controllers via a communication medium such as OpenFlow. It consists of three fundamental parts, i.e. OpenFlow protocol, secure channel and flow table database. Packets information and forwarding is performed with the help of a flow table. An SSL or TLS channel between the controller and switch is termed a Secure Channel. These protocols help in the management of communication.

Application layer is the uppermost layer that deals with the end-user. It comprises one or more applications, each of which can consume complete information about networks provided by one or more SDN controllers. Network necessities of SDN Applications (programs) are programmatically and directly intercommunicated to the SDN controller through Northbound Interfaces.

Southbound interface and *Northbound interface* are the interfaces used for communication and other objectives. The northbound interface is the interface between the SDN applications and SDN controllers and would be executed generally through REST APIs of SDN controllers. At the same time, the Southbound interface facilitates the communication between SDN controllers and the nodes in the Infrastructure layer of networks elements and would be generalized through southbound protocols. The Northbound interface (also known as Controller Application Interaction) links the controllers to the upper plane. It allows for the direct expression of network behaviour and the delivery of abstract network views. The Southbound interface, on the other hand (also known as the Control to Data-Plane Interface in other publications), links it to the lower plane and enables programmatic control of all event alerts, forwarding operations and statistics reporting. Unlike the Southbound interface, which has well-defined protocols like ForCES and OF (OpenFlow), the Northbound interface does not have any defined standard since that will be decided according to the logic of the SDN Application.

Hybrid SDN: SDN deployment is both time-consuming and costly. The approach, according to [2], is to install a small number of SDN-enabled nodes. It will build a network using SDN and existing network equipment. We will eventually replace all available network equipment with SDN-capable ones. Hybrid SDN is the name given to this sort of network. Hybrid SDN can be advantageous in a variety of ways, including deployment of a complete SDN is expensive; hybrid SDN can still have some of the advantages of the SDN architecture without a full SDN deployment; efficient management of scenarios where two SDNs are linked via a general network device; conventional routing protocols are also efficient for some tasks; easy evaluation; and so on.

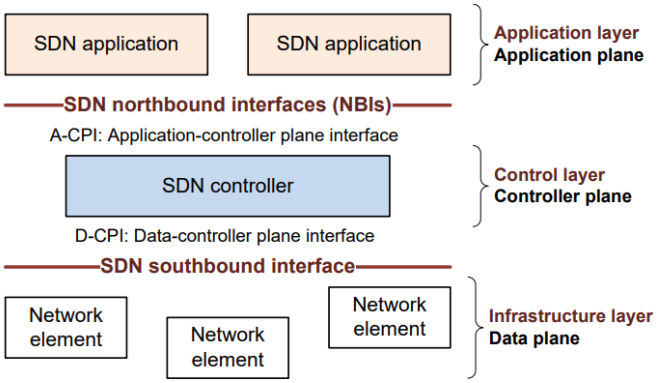

Fig. 1. Architecture Summary of SDN .

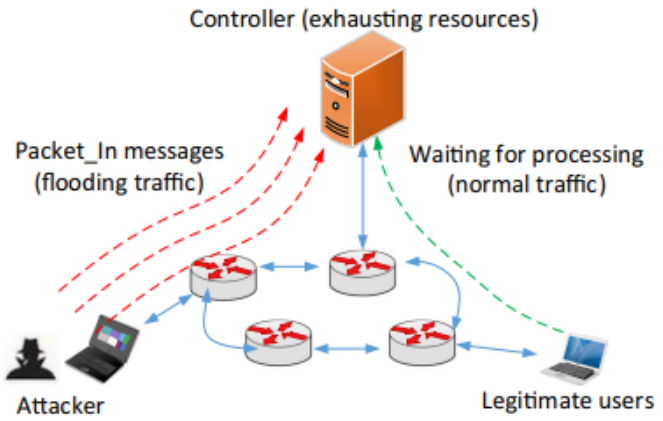

Fig. 2. Dos/DDoS attack on Controller.

SDN is most generally connected with the *OpenFlow* protocol, designed for Ethernet-based networks. The ONF organisation designated it as a generally acknowledged standard protocol. It accomplishes advancing actions cognitively as needed. An OpenFlow switch contains many flow tables, each of which contains a large number of flow entries.

II. Related Work

Recent surveys on SDN security tell a lot about the types of attacks that are frequent. Many authors have classified them in different types and evaluated the performance of network.

The authors in [3] discuss eleven distinct works that are based on SDN security. They talk about the 4D Project developed from the original work "A Clean Slate 4D Approach to Network Control and Management", OpenFlow Security, Kandoo, AVANT-GUARD, FRESCO. They also talk about Traffic Anomaly Detection. OpenFlow security was their main focus.

Authors [4] have briefly explained the Security Challenges in SDN. They have classified them as hardware-based challenges and protocol-based challenges. Controllers might be the primary target for the attackers or the OpenFlow protocol.

Authors in [5] have mentioned security issues and attack objects at different layers of SDN, i.e. Data Layer (Legality and Consistency of flow rules, DoS/DDoS attacks, Side-Channel Attacks, Authorized Authentication), Application Layer (Authorized Authentication, Flow Rules Conflict, Malicious Application), Control Layer (Poor Controller Deployment, Hijacked/Rogue Controller, DoS/DDoS attacks), Southbound Interface (OpenFlow protocol), Northbound Interface (standardization problem), Southbound interface (Communication security). Further, they have mentioned the causes of typical security problems in SDN and explained the existing defence technology for all these layers.

The authors at [6] and [7] also mentioned various security threats to SDN, their causes and discussed typical countermeasures that can be taken against the threats.

In [2] author give the general overview of the security related to the hybrid SDN networks.

We shall talk about the recent trends in security measures of SDN that use various recent technologies like Machine Learning, Blockchain and Firewall based techniques. We shall also discuss the various measures taken to detect and prevent DDoS attacks on SDN.

III. Frequent Attacks

A. Types of Attacks

In [8] there is mentioning of various types of attacks. For example:

Application Layer: flow rules insertion/conflict, malicious code, authorized authentication;

Control Layer: DoS/DDoS attacks, attacks from applications, man-in-the-middle attacks, hijacked controller;

Data Layer: side-channel attacks, approved authentication, flow rule consistency/legality Some of the attacks are briefly detailed in [9], as well as their classification according to the SDN layers. The attacks can be *Control Channel* specific attacks, which include attacks on protocols (e.g. OpenFlow), and*Data Layer* specific attacks, which are directed at network devices; *Control Layer* specific attacks, which are primarily directed at the SDN control and application layer.

B. Types of DDoS Attacks

In a network, the availability of a network device is very important. Hence, *DoS and DDoS attacks* are the most important aspects to be considered. The main attack mode is to impose excessive load on the controller, which reduces the availability of SDN resources to legitimate users.

There are various types of DDoS Attacks.

1) Control Layer DDoS Attacks: In this attack, the attackers' main target is the control plane. There are many methods of launching this attack. Some of them are attacking southbound API (OpenFlow), the controller and northbound API. Many conflicting flow rules from many devices will create much unnecessary traffic, and the controllers will be busy servicing unnecessary requests occupying a high bandwidth which may cause DDoS attacks on the control plane. Countermeasure would be to analyze the characteristics of traffic flows stored in the OpenFlow Switches or use of framework FloodGuard.

2) Infrastructure Layer DDoS Attacks: Attackers may target two things, i.e. switches or southbound interface. The packets with only header information require memory in the nodes. So, the attackers can release many such packets to flood the node memory. Node's memory is a bottleneck due to the cost factor. In order to save the fake packets, the required packets of the actual data might not get stored. Attacking the southbound interface, i.e., protocols like OpenFlow, is a common practice of the attackers. It usually breaks the entire flow of the network. Countermeasures would be to use the tool FlowVisor which acts as an agent between switches and the controller and rewrites the rules. Another measure could be the use of an intrusion detection system or Virtual Source Address Validation Edge (VAVE)

3) Packet Flooding: This type of attack is where the attackers generate a vast number of network flows and send them to the controller. As a result, the controller resources will be consumed by false data packets, and the controller could move to an unpredictable state.

C. Control Layer Specific Attacks

It is the most important component of SDN and the weakest link in the security chain. An intruder who takes over the controller will completely control the network.

1) Control Message Manipulation: Switch Table Flooding, Switch Identification Spoofing, and Malformed Control Message Attacks are standard hacking techniques used in this mode. The attackers attempt to modify control messages, and when the controller attempts to parse the message, it begins to work unexpectedly.

2) Hijacked Rogue Controller: Unauthorized programs can hijack the controller via upper-layer applications, leading genuine user requests to be refused [5].

3) Poor Controller Deployment: A multi-controller deployment scheme constantly reassigns failing controller loads to another controller, and if the controller is deployed incorrectly, it can easily overload the controller load beyond its capability, resulting in a failure.

4) Man in the middle Attack: Its fundamental principle is to place an agent node between the source and destination nodes, which is used to intercept and change communication data without being discovered. Session hijacking, DNS spoofing, port mirroring, and other specific attack methods are listed in [7]. Countermeasures include establishing a secure link between the controller and the switches (TLS) and employing a validation tool such as FlowChecker.

D. Application Layer Specific Attacks

It is made up of numerous types of applications. The flow rules can be defined by OpenFlow applications, security service applications, or other third-party applications.

1) Authorized Authentication: There is a lack of sufficient trust assessment and management mechanisms to validate the application's security. Attackers may use flaws in the application's role authentication and access control, as well as vulnerabilities, to construct malicious flow rules.

2) Flow Rules Conflict: Because of the wide variety of third-party open-source applications in SDN, there is a higher degree of coupling that may seem to compete, cover, or conflict.

3) Malicious application: SDN will suffer from unpredictable damages as a result of a malicious application. It consists of viruses, Trojans, and worms that can attack SDN via software agents with minimal effort.

E. Data Layer Specific Attacks

It comprises switches and other fundamental devices that are primarily concerned with data processing, forwarding, discarding, and collecting status information.

1) Authorized Authentication: It lacks a reliable authentication process between essential devices and the controller. Legal data packets may be manipulated or discarded by a malicious switch. If the switch is established without authentication, it might be operated by a hostile controller. Countermeasures include using the AuthFlow tool, which authenticates the host identity using a RADIUS server.

2) Legality and consistency of flow rules: Flow rule legality refers to the injection of malicious or inaccurate flow rules. Flow conflicts, inconsistent rules, and non-synchronized flow rules among switches can occur throughout the creation, release, and update processes. Countermeasures include the adoption of the NeSMA and Flowvisor frameworks.

3) Side channel attacks: Using side-channel assaults, an intruder can deduce network-related flow table knowledge by measuring the execution time of a particular type of data packet. It has the potential to indirectly trigger additional attacks.

F. Security Issues in interfaces

Southbound interface attacks are primarily triggered by protocol leaks such as OpenFlow. It is vulnerable to data leakage, controller spoofing, eavesdropping, and other threats. Because the interaction between the control and application layers is more brittle, the northbound interface faces a standardisation difficulty. Because of the diversity and continual updating of SDN applications, there are no uniform ways of permission and authentication.

IV. Security Measures Taken

A. Against DDoS Attacks

The first thought that comes when we talk about SDN security is securing the systems from DDoS attacks. Authors propose various techniques for the same.

The authors of [10] present a *optimised weight voting based ensemble model for detecting and mitigating* such threats in an SDN context. It employs six base classifiers, including two SVMs (Support Vector Machines), two random forests, and two gradient booster machines distinguished by hyper-parameter values. SVM is utilised due to its excellent generalisation capacity and resistance to over-fitting. A hyper-plane

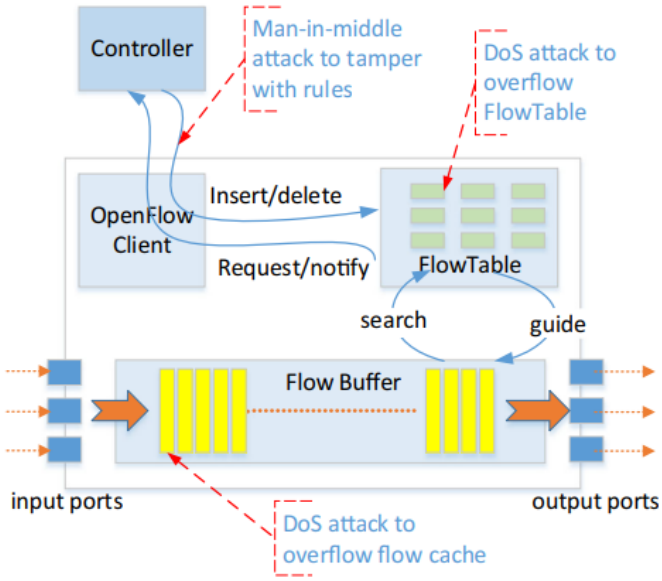

Fig. 3. Security Threats of OpenFlow Switches.

is a collection of $overrightarrowz$ points that satisfy the equation

$$\overrightarrow{\omega}\overrightarrow{z} + b = 0 \tag{1}$$

where $\overrightarrow{\omega}$ and b are the weight vector and bias, respectively and $\overrightarrow{z}$ is the point. By establishing hyperplanes, SVM classifies data into different classes. Out of an overwhelming number of hyperplanes, the one with the maximum distance between the closest distance in the two groups is regarded as the best. Random forests are also heavily used, but they lack accuracy because of their low bias and high variance. They integrate bagging with random variable selection. Gradient boosting is a strategy for creating an influential committee with increased effectiveness than single members by combining the findings of a large number of simple predictors. It could be used for regression as well as classification.

The authors propose an approach with some phases, i.e. learning phase, detection and mitigation phase. In the learning phase, they used DDoS Evaluation Dataset for training purposes. A novel hybrid metaheuristic optimization technique called "BHO" finds the best set of weights using a dynamic fitness function which is

$$1 - Accuracy = \frac{\alpha F^- + \beta F^+}{T^+ + T^- + \alpha F^- + \beta F^+} \tag{2}$$

where F^+ is False Positive, T^+ is True Positive, T^- is True Negative, and F^- is False Negative. α and β are to assign penalty weights to F^- and F^+.

The algorithm consists of five steps. The size of the population, the number of groups, the problem dimension, the lower and upper bounds of solutions, the maximum repetitions, and other parameters are all inputs to BHO. Finally, using the greedy selection method, the population will be passed to the worst agent handling module, where the N_w worst solutions will be chosen and substituted with randomly generated solutions. This phase aids in breaking out from the local optima trap. The findings from each base learner are gathered during the detection and mitigation stage. To identify a new flow as malicious or normal, a weighted majority vote of the findings acquired from each base learner is utilised as a final prediction. The overall estimate is

$$majority[W_1.SVM_1, W_2.SVM_2, W_3.RF_1, W_4.RF_2, W_5.GBM_1, W_6.GBM_2] \tag{3}$$

In [11] authors propose an entropy-based technique followed by Machine Learning. An introductory module based on information entropy is utilised for detection, followed by Machine Learning with a stacked sparse autoencoder. As the dispersion of a random variable's distribution rises, the information entropy increases. Entropy is defined mathematically as

$$\Gamma(z) = -\sum_{u=1}^{M} (\frac{\eta_u}{\Phi}) \log (\frac{\eta_u}{\Phi}) \tag{4}$$

where the probability of outcome x_u is represented by η_u/Φ and total number of samples in z is presented by $\Phi = \sum_{u=1}^{M} \eta_u$ Auto Encoder reconstructs the input data using the output of a neural network. To correct the model parameters, it uses the back-propagation gradient descent process.

In [12] a mitigation approach using *Deep Neural Networks* is suggested. Usually, rate-limiting logic is implemented as a defence mechanism to protect a system from excessive service.

There are several techniques for measuring and limiting rates, i.e. *Token Bucket* (Keeps a running and increasing use budget as a token balance and reduces the count whenever a service request is made.) If the bucket is empty, it signifies the limit has been reached, and no more queries may be processed.)

Leaky Bucket (The rate is limited by the quantity that can leak out of the bucket, similar to the token bucket.) It is assumed that the system has finite capacity and that any request that exceeds it is regarded as a leak and is rejected.)

Fixed Window (sets a restriction on how many requests may be made in a given amount of time) *Sliding Window* (has all of the advantages of a fixed window, but the sliding window of time smooths out the bursts.)

Deep Neural Networks are created by stacking neural network layers and the width and depth of smaller structures. They concentrated on lowering the loss function by dynamically modifying the authorised number of new flow queries in each host where a sliding window-based rate-limiting method is implemented. A weight specifies the maximum number of fresh flow queries a host can transmit to its edge switch during a cycle. At the end of each cycle, the weights are adjusted using two reductions (The first is computed using a training model that predicts the number of received PacketIn messages at a controller and then compares it to the available queue capacity using an MSE-based loss function. The second is obtained by evaluating the disparity between the currently available memory of a switch's flow-table and the amount

of PacketIn messages to the controller for processing.). The controller declares a new permitted number of flow queries to the host using OpenFlow messages when the weights are modified. Then the controller checks whether the changes after the new policies remain within a particular range. If it falls within the acceptable range, the host is regarded as usual; otherwise, it should be prohibited. The controller stops the host by adding a new default rule to the flow table that causes all incoming flow requests that do not match to be discarded immediately.

B. Machine Learning Techniques

Following our discussion of DDoS detection and mitigation strategies, which incorporated ML and Deep Learning techniques, we will go over more ML techniques and how they are applied in SDN security.

In [13], the authors have proposed an attack pattern prediction using ML techniques. ML algorithms are trained using traditional network threat intelligence data to identify potentially malicious linkages and probable attack targets. Based on *Conventional Network Data (DT)*, *Bayesian Network (BayesNet)*, *Naive-Bayes*, *C4.5* and *Decision Table* are used to predict the largest host that will be attacked. Bayesian Network computes probabilistic interactions between various interest variables. The random variable and its directional edge for each node are reflected in the resultant graph. Native-Bayes utilizes Bayesian Theory which employs the training samples to predict the sort of unidentified samples based on the previous results. Native Bayes divides the instruction set as a choice vector and attribute vector. C4.5 utilizes the decision making tree in common. It can handle extremely noisy data. In Decision Table, the primary learning phases were:

1) It selects an attribute modelled on which to evaluate a logical check.
2) A part of training data for the child node is extracted from the experimental results.
3) All the child nodes run the methods iteratively.
4) A node is represented by a leaf depending on specific concluding principles.

The experiments show that the overall accuracy is often more than 90%. In this study SDN, and NFV (Network Function Virtualization) are combined, and the proposed network was named Software Defined Network Function Virtualization (SDNFV). The suggested stateful firewall was set up to identify and prevent Denial of Service (DoS) attacks. The proposed methodology's main characteristics were

1) The use of traditional data to train ML-based applications
2) The use of the trained design to detect possibly susceptible hosts and establish the safety rules in the VNF (Virtualized Network Functions) module according to the predictive performance of the Machine Learning algorithm.

The design is trained using traditional data and then is used to determine susceptible hosts. Here,

$$Accuracy = \frac{n_{ap}}{n_t} * 100 \tag{5}$$

where n_{ap} is the number of attacks correctly predicted and n_t is the number of attacks in total. The choice of dataset affects the accuracy of the prediction. The greater the data variability, the greater the likelihood of incorrect prediction is. Using the prediction results of machine learning models, it is possible to define the security flow rules for stateful firewall VNF to avoid unauthorized clients from entering the network.

The authors in [14] have given an evaluation of the ML techniques discussed above (e.g. Naive Bayes, SVM etc.) for SDN security. They tested approaches such as Decision Tree, Naïve Bayes, Support Vector Machine and Logistic Regression. The experiment was a simulation-based experiment run on a Core i7 machine with a 1.80 GHz processor, 16 GB of RAM and 512 GB of SSD on Ubuntu 18.04. To create the testbed with Mininet, a remote controller with a tree-based topology and a depth of two is employed. Open vSwitches with POX controller and OpenFlow are employed. According to the data, SVM outperforms other algorithms with an accuracy of 97.5 per cent, a precision of 97 per cent, and a shallow error rate of 2.5 per cent. Except for Logistic Regression, all other methods have a 96 per cent accuracy.

C. Firewall Techniques

According to [15], a Firewall is a simple security technique that manages communications between distinct trust zones. The purpose is to make all networks flow through the firewall and allow only allowed traffic based on security strategy to pass, and and store logs because the firewall is impenetrable. Using a firewall approach, the following strategies are proposed.

1) Attempts are also made to reduce the overhead on the SDN firewall.
2) It is critical to distribute the load throughout the number of firewalls.
3) The stage concentrate on the control plane to govern the entire network, is more extensive and will include a slew of new features. These adjustments must streamline and speed networking operations.
4) Provide good flexibility and ease of configuration of firewall objects based on user needs and demands.

The OpenFlow protocol is used to construct an intermediary firewall. Every host has its firewall. A firewall object connects the hosts to the network, and then OpenFlow passes control to the server and allows traffic to pass. The Firewall module adds firewall capability to POX. These techniques contribute to the security of SDN.

The authors of [13] explore ways for anticipating attack patterns by utilising a stateful firewall as a Virtual Network Function via ML, as discussed under ML approaches.

D. Blockchain Techniques

In [16], the authors introduce B-DAC, a blockchain-based architecture for distributed identification and fine-grained access control for the northbound interface that will let administrators manage and safeguard vital resources. Cryptography like hash, asymmetric encryption, and digital signatures are used in blockchain to ensure that data is kept secure and confidential. It has a public ledger that keeps track of all digital transactions between the parties involved. This ledger is duplicated and stored across the whole network of peer-to-peer nodes. As a result, transaction records can be accessed even if multiple nodes are down or unavailable. The consensus process ensures that several nodes control the blockchain's database. Before being uploaded to the blockchain, new transaction data need to be verified by all nodes. As a result, hackers will be unable to change ledger records because they will need to control numerous nodes to defeat the consensus process. Hyperledger Fabric, often known as chain code, is an open-source application of permissioned enterprise-grade blockchain for executing smart contracts. It consists of three types of nodes: peer, client-owned by various organisations and ordered. A Membership Service Providers are responsible for identifying all nodes (MSP).

Authentication-authorizing-accounting deficiencies, lack of tamper-proof and integrity & Single point of failure (SPoF), harmful flow rules insertion, exhausting controller resources, and information disclosure are all current difficulties in SDN. To connect to a Hyperledger Fabric, a participant must have a unique digital identifier saved in a wallet, according to B-DAC. It employs a random integer to strengthen the security of the signature given a key pair comprising the secret key and the public key. Before encryption, a random number is generated.

The primary purpose here is to keep the lines of communication between both the controller and the OpenFlow protocol safe. The B-DAC architecture examines the trustworthiness of applications inside a network based on their behaviours to enable trust establishment. Along with all the other relevant information, the authors offer a new field called *trust* index. With over-privileged tries or conflicts in flow rule installation from OpenFlow applications, the trust index's value lowers. Only registered applications with a token can communicate with the system to facilitate authentication. These tokens are created and managed by the Token Management Module. Furthermore, there is a Permission Module that checks whether the application's action is over-privileged and requires attention. The core function of accounting is logging, which involves obtaining and saving log entries in a certain format. This job is managed by a Log Module in the design. Transactions are any activities that have the potential to modify the value of assets in a distributed ledger (blockchain). The AAA scheme is deployed outside the controller to increase the system's flexibility and independence.

To interface with B-DAC, controllers and OpenFlow applications can use RESTful APIs. Authentication to B-DAC's REST API, Authentication to the OpenFlow application, Authorization, and Accounting are all part of the AAA system. To update the flow rule, numerous data are extracted after receiving a request. These flow rules can be compared to a predefined conflict policy by verifying the flow rule's validity (all of the parameters and their values are extracted from the flow rule and then checked to see if they are supported and if their values do not violate restrictions such as range, maximum value, and so on), and then identifying and categorising flow rule dispute(To determine whether a conflict exists, compare a target flow rule to a stored conflict-free flow rules database). B-workflow DACs are divided into nine parts, beginning with request sending and concluding with request serving. The workflow includes nine steps: authentication (validation), token allocation, request delivery, permission parsing, conflict detection, and request service. The B-DAC system ensures security features such as immutability, decentralisation, authentication, authorization, accounting, flow rule conflicts prevention, fine-grained control, and Trust Level management. By managing OpenFlow Applications, this solution improves the safety of the northbound interface in SDN. Using the underlying properties of the blockchain, this solution intends to protect the communication between SDN controllers and network applications, i.e. the northbound interface.

V. Conclusion and Future Work

In this survey, we revisited the architecture of SDN succinctly. We described the types of attacks that are very frequent on SDN and various types of security issues. We discussed the countermeasures to prevent those attacks. We also described the security measures using ML techniques, blockchain and firewall, and other related techniques. Furthermore, we found some specific systems implementation to detect and prevent specific attacks. SDN is one of the most promising technologies, and existing security solutions are not sufficient to completely mitigate threat issues in SDN. Future work could include exploration of Machine Learning, blockchain techniques and implementation of the ideas provided by the authors. Also, there is an extensive possibility for further studies in the emerging field of hybrid SDN networks and their security aspects.

References

[1] S. Rowshanrad, S. Namvarasl, V. Abdi, M. Hajizadeh, and M. Keshtgary, "A survey on SDN, the future of networking," *Journal of Advanced Computer Science & Technology*, vol. 3, pp. 232–248, Nov. 2014. DOI: 10.14419/jacst.v3i2.3754.

[2] R. Amin, M. Reisslein, and N. Shah, "Hybrid sdn networks: A survey of existing approaches," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018. DOI: 10.1109/COMST.2018.2837161.

[3] M. Coughlin, *A Survey of SDN Security Research, University of Colorado Boulder*, 2014.

[4] H. Zhang, Z. Cai, Q. Liu, Q. Xiao, Y. Li, and C. Cheang, "A survey on security-aware measurement in sdn," *Security and Communication Networks*, vol. 2018, Apr. 2018. DOI: 10.1155/2018/2459154.

[5] Y. Liu, B. Zhao, P. Zhao, P. Fan, and H. Liu, "A survey: Typical security issues of software-defined networking," *China Communications*, vol. 16, no. 7, pp. 13–31, 2019. DOI: 10.23919/JCC.2019.07.002.

[6] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "Sdn security: A survey," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7. DOI: 10.1109/SDN4FNS.2013.6702553.

[7] Z. Shu, J. Wan, D. Li, J. Lin, A. Vasilakos, and M. Imran, "Security in software-defined networking: Threats and countermeasures," *Mobile Networks and Applications*, vol. 21, Oct. 2016. DOI: 10.1007/s11036-016-0676-x.

[8] M. Iqbal, F. Iqbal, F. Mohsin, M. Rizwan, and F. Ahamd, "Security issues in software defined networking (sdn): Risks, challenges and potential solutions," Jan. 2019.

[9] T. Jose and J. Kurian, "Article: Survey on sdn security mechanisms," *International Journal of Computer Applications*, vol. 132, no. 14, pp. 32–35, Dec. 2015, Published by Foundation of Computer Science (FCS), NY, USA.

[10] A. Maheshwari, B. Mehraj, M. Khan, and M. Idrisi, "An optimized weighted voting based ensemble model for ddos attack detection and mitigation in sdn environment," *Microprocessors and Microsystems*, vol. 89, p. 104 412, Mar. 2022. DOI: 10.1016/j.micpro.2021.104412.

[11] Z. Long and W. Jinsong, "A hybrid method of entropy and ssae-svm based ddos detection and mitigation mechanism in sdn," *Computer and Security*, 2022.

[12] A. E. Kamel, H. Eltaief, and H. Youssef, "On-the-fly (d)dos attack mitigation in sdn using deep neural network-based rate limiting," *Computer Communications*, pp. 153–169, 2022.

[13] S. Prabakaran, R. Ramar, I. Hussain, *et al.*, "Predicting attack pattern via machine learning by exploiting stateful firewall as virtual network function in an sdn network," *Sensors*, vol. 22, no. 3, 2022, ISSN: 1424-8220. DOI: 10.3390/s22030709. [Online]. Available: https://www.mdpi.com/1424-8220/22/3/709.

[14] A. Ahmad, E. Harjula, M. Ylianttila, and I. Ahmad, "Evaluation of machine learning techniques for security in sdn," in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020, pp. 1–6. DOI: 10.1109/GCWkshps50303.2020.9367477.

[15] D. Satasiya, R. Raviya, and H. Kumar, "Enhanced sdn security using firewall in a distributed scenario," in *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 2016, pp. 588–592. DOI: 10.1109/ICACCCT.2016.7831708.

[16] D. Phan, H. Hoang, D. Hien, A. Nguyen, and V.-H. Pham, *B-dac: A decentralized access control framework on northbound interface for securing sdn using blockchain*, Nov. 2021.