Security Enhancement Techniques in Software Defined Network: A Review

Anishka Gupta[1], Student, Prof. Jayanthi P. N[2]., Assistant Professor, R.V. College of Engineering, Bengaluru – 560059

***Abstract*—**In recent scenario, most of the organizations are moving their network to wireless systems. Wireless network provides more flexibility, scalability and centralized controllability to the network. As the organization's geographical area expands, need for Software Defined Networking arises. Also, the security measures are critical in Software Defined Network (SDN) because the control plane resides on internet and hence prone to malicious attack. In recent years, many security techniques, protocols and policy enforcement have been proposed. This paper reviews some of the recent past five years work in the domain of security of Software Defined Network.

***Index Terms*—Bayesian network, fuzzy logic, OpenFlow controller, OpenSec, software defined network**

I. INTRODUCTION

MANY leading IT-Companies such as Juniper, Cisco, IBM, VMware, and Dell have developed their own SDN policies and strategies [1]. SDN provides network virtualization by splitting the data plane and control plane of the network over internet. It enables centralized configuration and control of the network [2]. Additions of extra end-devices have also become simpler in SDN. Controllers are placed on a server or virtual machine, from where it employs network rules on the network devices and handles the network traffic and security [1 - 4]. The different security techniques based on user requirements from various business aspects can be enforced onto the controllers. Most of these techniques are based on OpenFlow protocol [5]. The OpenFlow protocol is widely used for communication between controllers and other network devices. Different protocols have been added to the OpenFlow protocol for security enhancement purposes.

Rest of the paper is designed as follows. Section II gives an overview of SDN architecture and OpenFlow protocol. Section III discusses the state of the art technologies that have been introduced in recent past. Section IV draws conclusion from different techniques reviewed.

II. SOFTWARE DEFINED NETWORKING USING OPENFLOW

In this section the SDN architecture is discussed in brief. Next, the OpenFlow protocol which is widely used for communication between devices is discussed.

A. SDN Architecture

SDN architecture is shown in figure 1, has three main layers named as data layer, control layer, and application layer when seen from bottom to top. End devices are placed in data layer which is responsible for bandwidth virtualization and data forwarding. Multiprotocol Label switching (MPLS), internet, 4 Generation (4G), broadband are some example of links present in the network. Bandwidth virtualization integrates various wireless connections that support one place into a data pool that is open for both devices and providers to allow maximum use of the bandwidth resource. Information propagation consists of a centralized collection of network components (primarily switches) responsible for transmitting data utilizing the bandwidth offered by bandwidth virtualization. Devices in data layer receive command from controller in upper control layer through interface protocol such as OpenFlow. Control layer, being the heart of SDN, have many network function to perform. For instance, network management gives a comprehensive view of the network traffic engineering, whereby the later determines an optimum coordinating method for network execution. QoS ensuring requires caution during data transfer to meet the program requirements. The application layer enables network operators and network engineers to announce their individual network needs by network speech and device speech, all of which may convert high-level specifications articulated as if in human language into compatible network configurations.

In the physical architecture of SDN, the data layer are interconnected through set of SDN switches by physical links. A network controller device is a server or group of servers that relies on the network capacity, position and functionality. Specific applications are placed on top of the network controller.

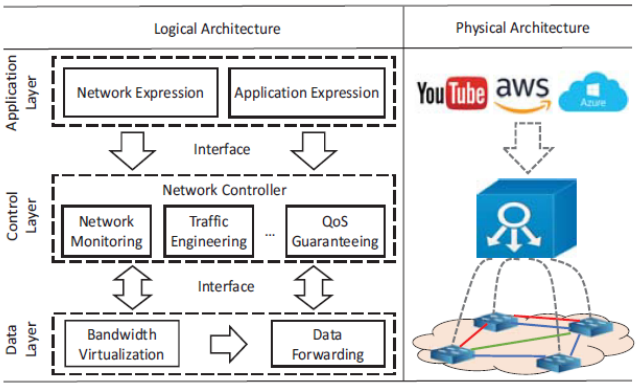


Fig. 1. SDN: Logical and Physical architecture [4]

B. *The OpenFlow Protocol*

As defined in [5], OpenFlow is proposed as the first common collaboration framework among control layer and the network layer of a software defined WAN system by the Open Networking Foundation (ONF). It offers a way to monitor a turn, without forcing vendors to share any of their devices' origin code. As shown in figure 2, OpenFlow primarily contains following three components: switch, channel and controllers.

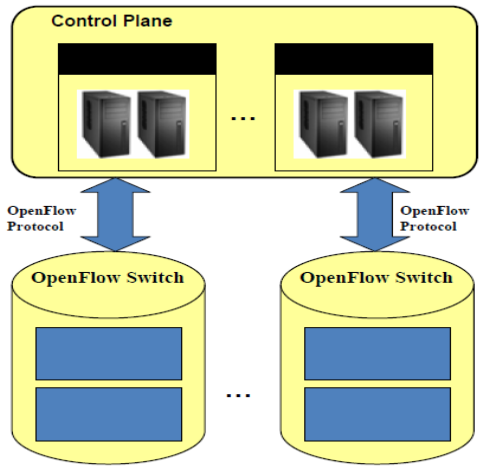


Fig. 2. Basic OpenFlow components and communications [5]

OpenFlow based switches are controlled by OpenFlow controllers utilizing OpenFlow protocol over a protected connection. A transferring channel contains multiple flow entries in a form of table that conduct the packet information searching and passing it ahead. In specific, such tables consists of a collection of flow inputs where these inputs include header areas, counts, and behavior. Header details have been required to align data with details including VLAN Identifier, IP address, origin and target ports, etc. Counters are primarily used to track message data, such as the number of messages, byte numbers, etc. Actions include guidance about how to handle and align packets in a river, such as moving them to a different port and sending them to a controlling device and also discarding the data.

OpenFlow channel are the connection point for interfaces between switches and controllers. The controller do the initialization, setup and management of the switch through this interface, receives transfer incidents, and sends out the packets. OpenFlow controller is accountable for the maintenance, distribution and updation of network device policies and instructions. This will decide whether to manage packets without legitimate flow entries and whether to add or remove flow entries through the protected path, it can manage the switch flow table.

OpenFlow transfer can include several flow tables, whereas several flow inputs may be given for each flow table. The switch would be able to look up the flow entries and take forwarding decisions according to the flow table for incoming packets.

III. STATE OF THE ART TECHNOLOGY

In this section, various state of the art technologies proposed in recent years are discussed. These techniques include machine learning, probability concepts, networking concepts and are implemented at control layer. The network emulator, such as Mininet, is used by authors for implementation.

A. *Secure OpenFlow Controller using Bayesian Network*

In [6], the authors focused on securing OpenFlow controllers using the preliminary packet filtering rules and the Bayesian Network (BN) classification of attacks. The new security features were implemented by the authors into the current OpenFlow protocol based controller. Packet filtering judgments have taken upon the filtering regulations and Bayesian representation of the network. Figure 3 displays the methodology between the OpenFlow protocols based controllers and switches, which is discussed and used in [6].

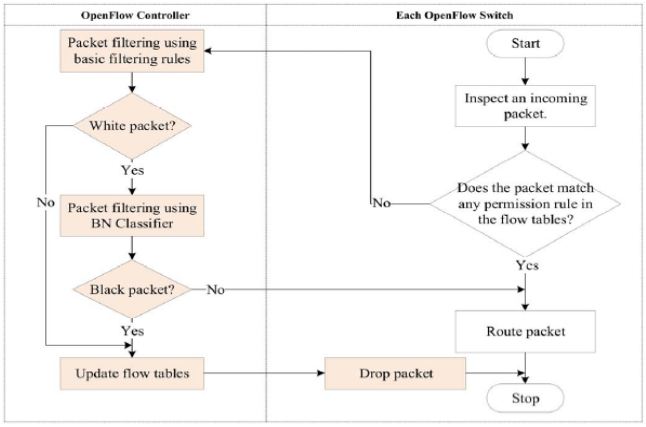
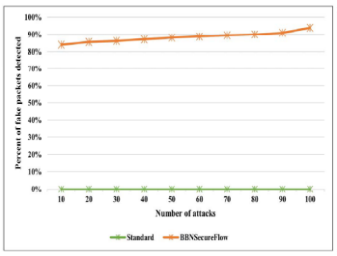


Fig. 3. Process flow of secure Openflow controller [6]

When a switch received a data segment, the switch tries to identify that segment in its available flow table to a flow entry. When no flow entry matching this flow is found, it will send the packet as a packet-in (PI) for careful examination and management to the OpenFlow controller. First and foremost, few simple rules are used to filter the packets and make the decisions to drop illegal packets, except the data these packets can include packet header detail, encapsulated protocol, ports of origin and of destination for transport layers, ICMP message sort and sending and receiving packet frameworks. If the packet verifies each rules for the filtering of packets, the packet is white (WP). Or Else, the packet is considered black (BP) and such packets will be discarded after flow tables are updated.

Then, the Bayesian network (BN) is used to inspect the packets that have passed the first filtering rules (white packets). After that, these packets are categorized in following two types: OpenFlow White Packet (OWP) and OpenFlow Black (OBP) packets. Where the likelihood of recurring destination IP address is more than or equivalent to 0.8, then this is OBP. Other than that, it would be OWP.

Implementation of this technique is done using Pyhton script in emulator Mininet. Evaluation shows a fake packet filtering success rate of 90%. The results show that the success rate of fake packets detection and discarding them is between 84.00% and 94.00%, which gives an average rate of 88.42%. This proposed method's detection rate is much better than the original OpenFlow controller as shown in figure 4. The findings indicate that the proposed controller is quite similar for the processing time as shown in figure 5. Hence, new techniques can be implemented with similar processing time. Though the processing time of proposed method is slightly higher than standard. It can be inferred that the proposed method would offer adequate protection and efficiency without having to incur a lot of overhead processing.



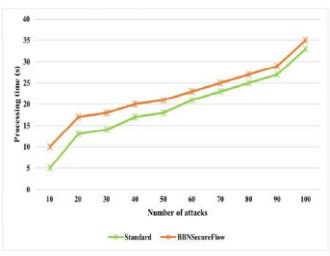


Fig. 4. Detection Accuracy [6]Fig. 5. Detection Performance[6]

B. Authentication Protocol for SDN

Allouzi and Khan [7] propose a further protocol, implemented on OpenFlow switches to deny access without valid digital credentials. This paper is about building trust among strangers. Strangers are the devices here. The presented protocol is known as SafeFlow-an automatic protocol for SDN trust negotiation. Once a controller starts communication to a switch, the controller verifies the OpenFlow switch as per the already defined protocol which specifies the OpenFlow switch to only treat as trusted for the defined data. SafeFlow protocol is the enhancement of the OpenFlow handshake convention that incorporates faith management to provide security inside the SDN network. The suggested authentication solution is focused on the sharing of digital certificates to slowly create verification. The electronic attributes include electronically signed documents by an issuer of credentials concerning a credential owner. A certificate uses user-title / meaning duo to identify multiple characteristics of the recipient. The X.509v3 certificates are only an instance of a token which is electronically signed. Considering that digital passwords will also contain confidential material, an access management program must track passwords disclosure.

The SafeFlow protocol (figure 6) sets up one new message form, called the negotiation message. This is built to enable SDN authentication among OpenFlow switches and controllers if an OpenFlow based switch demands access to a critical resource from the OpenFlow controller. The controller will send the switch a message on Negotiation Request. The switch replies with the other Negotiation Request response which contains the desired service whether it is available or vulnerable, or the controller can combat requests with a better resource.

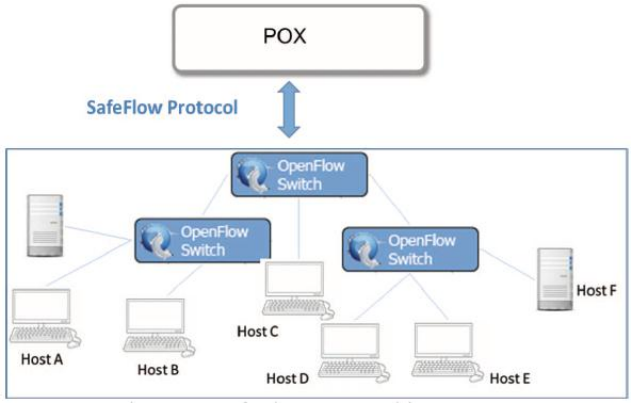


Fig. 6. SafeFlow protocol in SDN [7]

The switch and controller of the OpenFlow protocol mandatorily concur on a policy to inbuilt trust, in order to conduct a successful trust negotiations. When utilizing trust negotiation, there are two methods to enable authorization: constructing a layer above that of the OpenFlow protocol or incorporating trust negotiation into the OpenFlow access control transport layer security (TLS). The SafeFlow protocol is a case in point for latter approach. SafeFlow is successfully executed in Python. However, the authors haven't done performance evaluation/comparison based on any network parameters.

C. Policy Based Security in SDN

A security platform built on OpenFlow which enables cyber security users to build and enforce privacy policies that are written in a human-understandable language, known as OpenSec is proposed by Adrian and Byrav [8]. Using OpenSec, person may characterize a flow based on the fields described in matching table to OpenFlow, identify what protection policies are to add to the flow (transmission analysis, network monitoring, spam detection, etc.) and determine protection rates that decide how OpenSec responds when attacks are found. The key aim of OpenSec is to provide a beings-friendly, responsive and self-employed platform for security. The three architecture specifications of the proposed system are described: transfer middleboxes far from central data path, respond to security incidents without human intervention and build a basic policy specified language.

The very prior aim of OpenSec is to push middleboxes far from choke points of network from which all data flows. Other than that, such machines would be placed out of key route between the local network and the Internet, which would serve as protection entities that are accessed only by the data packets that requires processing. The OpenSec will build guidelines for re-routing traffic dynamically utilizing a smarter, control plane focused on OpenFlow. The next aim of OpenSec is to allow automatic response to privacy events. If a middlebox senses unusual data packets it may send a warning. A network operator usually gets such warnings, and then determines whether to respond to them. With OpenSec this mechanism is implemented in such a way that either the system prevents data packets, or merely tells the user to the observed attacks. The third purpose of OpenSec is to include a common language for policy requirements to help subscribers to divert traffic to the middleboxes and allow for automatic reaction

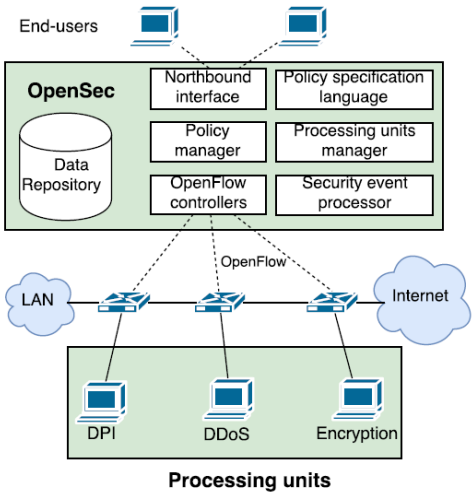


Fig. 7. The OpenSec framework [8]

To evaluate OpenSec performance, three observational configurations are presented that were implemented on the test platform of GENI. The execution period is of several milliseconds. The time necessary to block attacks is almost similar, independent of the rate of traffic. Before reaching destination 95.5% of flows are stopped and almost only 1% of the attacks could not be stopped by the network. The implementation of OpenSec is compared to[9] and [10]. It can be concluded that the improvements in security by the proposed work are human readability have increased as the policy specification language is simpler to understand, automated response to privacy warnings is possible. In contrast to the improvements, making routing conclusion to find the best controlling unit is not present in OpenSec.

D. Information Security Management for SDN

The proposed method by authors in [11] considers the soft computing-based information protection management framework algorithm and the SDN intrusion detection framework intrusion detection system (IDS) implementation was introduced, consisting of a module for data compilation, analysis and decision taking. This system includes: assessing information system security levels, managing information security risks, and detecting and preventing intrusion. These modules were implemented as Beacon Controller applications in Java. Figure 8 shows the algorithm and the behavior carried out at each of its points.

Device evaluation was performed on a few of the major networks protection issues-identifying devices involved in harmful network scans. In Java, jFuzzyLogic library has been used for software implementation of decision making algorithms. Fuzzy analysis were carried out through Mamdani methodology that presently seems to have the vast realistic implementation in Fuzzy Modeling issues since this algorithm focuses on Fuzzy reasoning and prevents unnecessary computation. Authors have used Fuzzy Control Language (FCL) as standard.

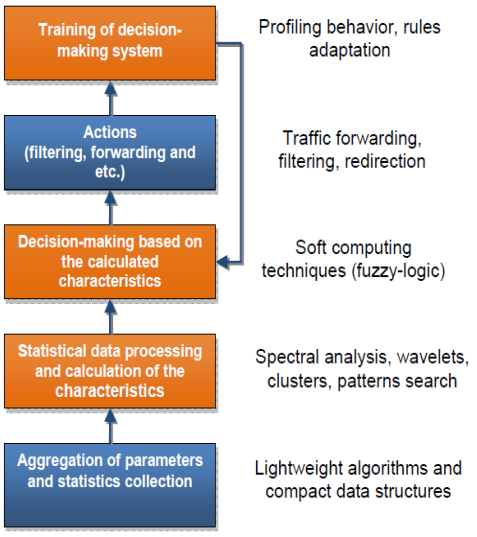


Fig. 8. Proposed Algorithm [11]

OpenFlow emulation mininet is used to build software-defined network, which offers a quick prototyping workflow. Single network switch and a few device attached to the OpenFlow switch are emulated in mininet, as is a machine operating the Beacon controller. For flow creation, two hosts are chosen to establish a TCP or UDP link to distinct other hosts. In a simulated world, a testbed was built too. In this evaluation, each device launched malicious packets at various rates, at three small (0.1/1/10 pps) and two large (100/1000 pps). Every activity is simulated for two minutes. Attacks were intermingled with daily consumer information. And there were about 1700 contacts, and about 520,000 packets were transferred.

As a consequence, the suggested program detected 95% of the attacks at 1.2% fake positives. A significant advantage of the proposed method is the amount of code lines shortened by 20-30 percent, also the ability to combine different remote modules and collections efficiently, thus significantly simplifying the execution of algorithms and decision-making processes.

IV. CONCLUSION

As SDN is wireless network and is expanded in a large geographical area, it will always be prone to intrusions and malicious attacks. Thus, security enforcement policies will always be in research. The technique discussed above provides security from various attacks in SDN at different levels and hence all of such techniques can be implemented together to provide a robust SDN architecture. More intelligence can be added to the network using data analysis and introducing Deep learning/ Machine learning techniques in the network. In future, a controller can be dedicated for all security related issues in the SDN, such as security policies enforcement, controlling the flow and taking multiple actions according to the type of attack.

References

[1] Dacier, M. C.; et al, Security Challenges and Opportunities of Software-Defined Networking, IEEE Security & Privacy, Institute of Electrical and Electronics Engineers (IEEE) article, 2017, 15, 96-100

[2] Kreutz, D.; Verissimo, P. E.;Ramos, F. M. V.; Rothenberg, C. E.; et al, Software-Defined Networking: A Comprehensive Survey, Proceedings of the IEEE, Institute of Electrical and Electronics Engineers (IEEE), 2015, 103, 14-76

[3] Rawat, D. B. & Reddy, S. R., Software Defined Networking Architecture, Security and Energy Efficiency: A Survey, IEEE Communications Surveys & Tutorials, Institute of Electrical and Electronics Engineers (IEEE), 2017, 19, 325-346

[4] Z. Yang, B. Li, Y. Cui, Y. Xu and Y. Liu, Software-defined wide area network(SD-WAN): Architecture, advances and opportunities, in 2019 28th InternationalConference on Computer Communication and Networks (ICCCN), IEEE, Jul. 2019

[5] Li, W.; Meng, W. & Kwok, L. F., A survey on OpenFlow-based Software Defined Networks: Security challenges and countermeasures, *Journal of Network and Computer Applications, Elsevier BV,* 2016, 68, 126-139

[6] Sophakan, N. & Sathitwiriyawong, C., Securing OpenFlow Controller of Software-Defined Networks using Bayesian Network 2018, 22nd International Computer Science and Engineering Conference (ICSEC), IEEE, 2018

[7] Allouzi, M. & Khan, J., SafeFlow: Authentication Protocol For Software Defined Networks, 2018 IEEE 12th International Conference on Semantic Computing (ICSC), IEEE, 2018

[8] Lara, A. & Ramamurthy, B., OpenSec: Policy-Based Security Using Software-Defined Networking, IEEE Transactions on Network and Service Management, Institute of Electrical and Electronics Engineers (IEEE), 2016, 13, 30-42

[9] S. Shin and G. Gu, CloudWatcher: Network security monitoring usingOpenFlow in dynamic cloud networks (or: How to provide securitymonitoring as a service in clouds?), in 2012 20th IEEE InternationalConference on Network Protocols (ICNP), Austin, Texas, October 2012.

[10] S. Shin, V. Yegneswaran, et al,FRESCO: Modular Composable Security Services for Software-

Defined Networks, in Network and Distributed System Security Symposium(NDSS), San Diego, California, U.S.A., February 2013.

[11] Dotcenko, S.; Vladyko, A. & Letenko, I., A fuzzy logic-based information security management for software-defined networks, 16th International Conference on Advanced Communication Technology, Global IT Research Institute (GIRI), 2014