DoS Attack Prevention on IPS SDN Networks

Awatef Ali Yousef R.Fares*, Francisco L. de Caldas Filho*, William F.Giozza*, Edna Dias Canedo†, Fábio Lúcio Lopes de Mendonça* and Georges Daniel Amvame Nze*,
* Electrical Engineering Department, University of Brasília, Brasília, Brazil
†Department of Computer Science, University of Brasília (UnB), Brasília-DF, Brazil
Email:{awatef.fares, francisco.lopes, fabio.mendes}@redes.unb.br, {giozza, ednacanedo, georges }@unb.br

***Abstract*—The proposed solution addresses a security implementation for denial of service (DoS) attacks on software-defined networks (SDN) for which a combination of two technologies deployed on traditional networks that have been adapted to the SDN architecture is used. Implementing an intrusion prevention system (IPS) in an OpenFlow environment has emerged to enhance network security by analyzing data that travels through its structure and, together with this tool, the honeynet feature, executed through the command of an SDN controller. which orders packets identified as different from predefined transmission patterns to be forwarded to address a possible DoS attack.**

***Keywords*—SDN, OpenFlow, IPS, SDN controller, Security, Honeynet.**

I. Introduction

Software-Defined Networks (SDN) has been widely used in recent years as they offer many benefits over traditional networks including ease of deployment, flexibility, programmability, and scalability. As data usage and traffic continues to grow in real networks, the risk to information security increases, affecting multiple organizations making them vulnerable to potential DoS attacks through their computational resources. With the help of SDN technology, the security of this data can be provided through a centralized control point. In the SDN network, adapting new services and managing their structure, performed through a controller that manages routing rules for any connected network device to your infrastructure [1].

The most widely used protocol in SDNs is OpenFlow, through their use, tables are configured to assist in the management of all packet forwarding decisions that act in conjunction with central controller software that reaches all network assets [2], which makes managing your structure easier.

Regarding the security functions of SDN, it can be said that their centralized administration can provide a certain defense mechanism to prevent any DoS attacks [3]. As a DoS attack can make a server or network infrastructure unavailable, the OpenFlow controller does not specialize in analyzing abnormal traffic situations, such as those provided in current IDS and IPS, for this purpose IPS snort is used. Snort is a free software to identify intrusion in a data network developed initially by Martin Roesch, it performs real-time network TCP/IP traffic analysis, identifies packets that are not in the determined pattern and may suggest network security attacks [4].

SnortFlow is an OpenFlow-based IPS and security-focused protocol used in [5] for SDNs within virtualized infrastructures, where it is possible to detect intrusions and implement measures to prevent and correct these attacks. The results show that SnortFlow is ideal for security testing in SDN Networks. SnortFlow uses more complex syntax rules than other IDS appliances such as Snort and Suricata. These last two have a large community that contributed to the software development and the inclusion of new rules for the identification and mitigation of attacks. IDS and IPS are able to identify and analyze data that travels through the network by observing packets that have predefined standards. These devices can act to provide possible network intrusion detection solutions. SDN controller's programming capability to intervene by blocking or diverting malicious traffic should be used in conjunction with a honeynet (that implements IDS and IPS security mechanisms) to handled malicious traffic [6].

978-1-7281-2920-4/19/$31.00 © 2019 IEEE

According to [7] and [8], a honeynet is a network of honeypots designed to intentionally allow attackers to investigate, scan and exploit services considered vulnerable and, the attackers are unaware that they are in a mock environment.

The work developed in this document integrates an IPS / IDS system to analyze anomalous traffic on the local network, directing it to a controlled environment, where it should be analyzed. This article demonstrates that it is possible to monitor, treat and mitigate a denial of service attack on SDNs, combining security features. The proposed architecture consists of IPS and a honeynet managed by an SDN controller.

II. Background

A. Security in local area networks

Security in local networks usually is concentrated only on the Firewall or IDS. It is usually invested in perimeter protection systems to prevent external attacks, but it is forgotten that an attacker can unintentionally launch attacks from within the internal network by connecting an infected host to the network, or intentionally by promoting DoS attacks against servers or attempting to obtain information from which he or she is not allowed. Thus, safety features must be implemented in all layers. Local Area Networks (LAN) with more than 100 terminals are usually divided into three layers, namely access, distribution, and core [9].

The access network provides connectivity to end-users by connecting computers, notebooks, IP phones and Access Points (AP) among other network devices. User terminals are wired into high-port density switches and can be segmented into different virtual networks using Virtual Local Area Network (VLANs) [10].

APs are also part of the access network and allow the connectivity of mobile devices that implement the IEEE 802.11 protocol for Wireless Local Networks architecture (WLAN) [11]. Therefore, the primary role of the access network is to interconnect end-users with the LAN by utilizing high-port density switches and AP to interconnect mobile devices. Some functionality should be employed in the access network to increase availability and integrity.

Port Security [12] limits the amount of MAC addresses per port, preventing new switches from being added to user ports. This would, for example, prevent MAC Flooding attacks such that only trusted ports can respond to DHCP Request/Reply messages, and unauthorized DHCP servers from being installed on the network and would thus be able to self-report to the network gateway. Doing so would prevent Men In The Middle (MITM) attacks. These features, which previously relied on manufacturer support to include them in the switch data plane, can be programmed into the controller and implemented on any equipment, regardless of any manufacturer. Other layers two protocols of the OSI model such as Spanning Tree, which avoids network loop and ensures redundancy can also be included via the SDN controller.

One of the key actions to ensure security in any corporate LANs is to restrict access to any communication and data resources to their authenticated users. This can be done by using the IEEE 802.1x protocol, which limits the data protocols that can be transmitted by the host on the network until the user authenticates [13]. Authentication takes place with the Protected Extensible Authentication Protocol (PEAP) passwords and Lightweight Directory Access Protocol (LDAP) servers or using the Extensible Authentication Protocol-Transport Layer Security (EAP-TLS) digital certificate exchange, which authenticates both the machine and the user. This avoids the inclusion of non-corporate endpoints using their resources without proper authorization and makes it difficult for malicious attackers to access the internal network.

The distribution layer, presented in this paper, interconnects the different access subnets with the Core network. The use of end devices at the distribution layer can be useful to increase their availability, with Fist Hop Redundancy Protocol (FHRP) such as VRRP, which allows the creation of a cluster with different gateways spread out in this layer. The default gateway address configured on notebooks, personal computers, IP phones, or any other device installed at the access layer will be the Cluster address. On the other hand, the highest priority gateway will always respond as the default primary gateway. In case of failure of the main gateway, another previously configured gateway device will assume its role transparently. Concerning access security, security policies can be applied to limit user access to different subnets by using access lists on devices configured at the distribution layer. As an example, we may limit access for users of a company's research subnet to a company that wants to access data from the financial subnet. Thus, we would have several access lists to restrict any kind of routing between the two subnets. Layer 3 switches and routers can apply packet filters as complex as firewalls, allowing for even greater security of LAN access. Packets at this layer can also be parsed by any Network Driver Interface Specification (NDIS) installed in bridge mode, which can act in conjunction with Firewalls by applying filters to prevent anomalous traffic from being routed to any subnet.

For security purposes, packets targeted to enter the Core network will be routed to an IDS (Our Firewall device also implements a free open source network intrusion detection and intrusion prevention system (SNORT), as shown in Figure 1) and, those wishing to enter the internal subnets will be filtered for further security analysis.

Packets leaving the internal network should also be inspected by an IDS to identify anomalous situations, such as the use of disallowed applications, unauthorized users using the network, terminals generating an abnormal volume of traffic, or opening simultaneous connections without a closed connection state confirmation (as seen in Transmission Control Protocol (TCP) Flooding). By identifying such situations, an IDS may send an alert to the network administrator as well as to interrupt such data traffic. However, it would be safer to prevent the attacker's data flow by isolating him from the network, hindering him from having access to other network resources. Unfortunately, intrusion prevention systems are unable to reconfigure conventional switches, to disable the attacker's port or deauthenticate any legitimate user from the network. With the appearance of SDN networks, switch reconfiguration takes place centrally and integration between the controller and other network assets can be done using modern integration tools, such as the Representational State Transfer application programming interface (REST API), RESTConf and other protocols.

III. Related Works

Other works that propose to implement the use of security in SDN for virtualized environments are mentioned in this section where it is possible to get an overview of the possible solutions available in the market and their probable implementations made to date.

C. Jeong, et al. [14], proposes a scalable IDS intrusion detection system architecture for large SDN networks using a Kernel-based Virtual Machine (KVM). The KVM architecture is a virtualization system where one physical host machine is used to host and control several other virtual machines. The architecture proposed by the authors makes use of OVS switches, using the OpenFlow protocol and connected to each other through OVS tunnels in a virtual environment. For purposes of sample analysis of suspicious traffic, the authors used various IDS to perform a Deep Packet Inspection (DPI) technique on their network. The solution developed and presented in this work also used OVS switches for simulation purposes, but unlike [14], all malicious traffic is routed to a honeypots network.

Monshizadeh et al. [15], proposes a method to detect security attacks and malicious traffic early enough and to protect the network. They use centralized monitoring and intrusion detection system (IDS) to enhance any SDN, NFV,

and OpenFlow security. However, unlike our proposal, they do not divert the malicious traffic to a honeypots via an SDN controller.

In [16], Nam and Kim propose the implementation of a security mechanism without the use of additional hardware using SDNs. This mechanism consists of firewalls, network scanning systems, abnormal traffic detection, intrusion detection, and intrusion prevention techniques using OpenFlow technology. According to the study mentioned, all devices on an SDN network can act as a firewall by merely identifying packets that initiate a TCP session. OpenFlow has as one of its features a packet forwarding and discarding mechanism, which enables the deployment of any security devices in an SDN architecture environment. We propose in this paper, a method to identify security breaches using an IDS and SDN controller to block only anomalous traffic.

Andritsos [12] exposes some of the security issues that can affect a data network, including DoS attack, flow aggregation, access control, and insider disclosure. The proposed operational security requirements consider the prevention of unauthorized external access and reliability. We propose in this paper, a solution that also performs the prevention of attacks, however, using an IDS/IPS integrated with an SDN Controller (RYU's SDN open source Controller) and Firewall, to identify anomalous traffic.

The honeynet proposed in Kim and Shin [17] is implemented in a software-based/network-based system where the honeynet topology is designed to mislead and obtain more information about the attacker, such as its source IP address, and which data of interest is a target by the attackers. Unlike the proposal of Kim and Shin [17], our prototype uses a well-known honeypot system integrated into the SDN environment, directing anomalous traffic and generating alerts when a security breach occurs, and advising the network administrator to take any action.

In Kyung et al. [7], a HoneyProxy is proposed to monitor internal traffic with the help of the SDN controller. HoneyProxy uses a new connection management mechanism between different honeypots in the network to support honeypot transitions. To this end, a HoneyProxy-enabled SDN controller centrally programs the reverse proxy module that operates in three specific modes. The model proposed in this paper allows the use of more than one IPS, which will update the SDN Parent's flow table using REST calls.

IV. IPS ARCHITECTURE FOR SDN NETWOKS USING IPS

The proposed architecture implements an integrated security solution, with IDS identifying anomalous traffic originating on the local network and informing the controller which hosts are originating this traffic.

This architecture described in Fig. 1 demonstrates the arrangement of the elements used in the proposed network.

The IDS is positioned at the exit of the LAN network, monitoring traffic directed to the WAN and DMZ, so DoS attacks originating from the local network and directed to external servers are easily detected and directed to the honeynet network. The IDS notifies the local SDN controller by informing the source IP of the host that is generating unusual traffic. The host protocol promoting the DoS attack is also used for this matter.

Based on the information provided by the IDS, the SDN controller makes the following decisions:

- moves the host to a quarantined Virtual Local Network (VLAN), restricting its access to internal network devices and thus preventing botnets from contaminating other hosts, such as Mirai [18] malware that stood out in 2016, and was responsible for performing one of the most significant DDoS attacks to date, reaching traffic of up to 1.2 Tbps;
- directs anomalous traffic identified by IDS to a honeypot. This way, the system will log all attacker activity and send it to the administrator, providing clues of what the attacker intends to achieve.

Using the data obtained from the IDS logs and honeypot, the network administrator can confirm if the traffic is anomalous and, if so, makes the necessary corrections on the host. The components used in this paper are explained throughout this Section.

A. SDN Controller

One of the main benefits of SDNs is that it allows the separation of the data and control plane. Improvements can be continuously included in the control plane and can be passed on to any proprietary and non-proprietary equipment running the OpenFlow protocol.2

New features are added without the need to purchase new equipment or firmware upgrade. SDN switches can coexist with conventional switches; features such as spanning tree, link aggregation, and port mirroring can be easily added to the controller.

The controller chosen for this solution is Ryu [20] for its ease of setting up new functionality, extensive documentation, and the ability to deploy the solution in environments using Mininet (Which provides a virtual testbed and development environment for SDNs). Ryu also allows Snort integration using RESTFull calls.

Ryu software's controller can be written using Python 3. A script has been developed to perform network security settings from abnormal events received by Snort via REST calls.

Upon detecting an anomalous situation, Snort generates an event, informing the source IP that generated the attack. With this information, a script makes a REST call to the controller that redirects the malicious source IP address to the honeynet. The administrator receives an alert informing that an event is detected on the network and needs an investigation. The source IP is interrupted by transmitting packets on the network for an interval determined by the administrator.

B. Honeynet

Two honeypots servers were created in the test environment, directing malicious traffic to the applications installed on these

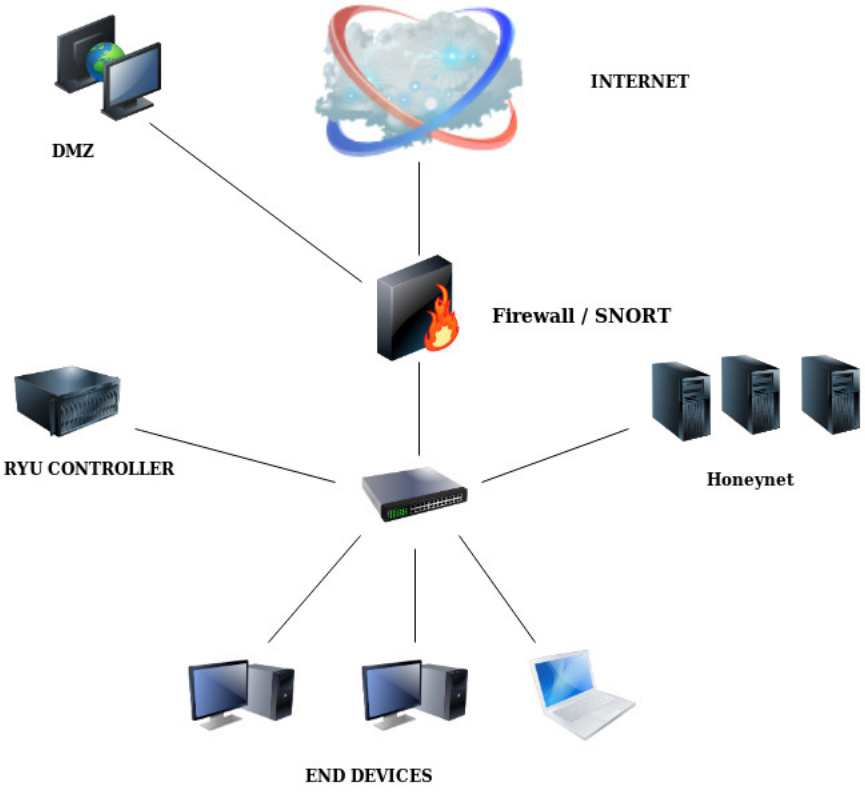

Fig. 1. Proposed topology.

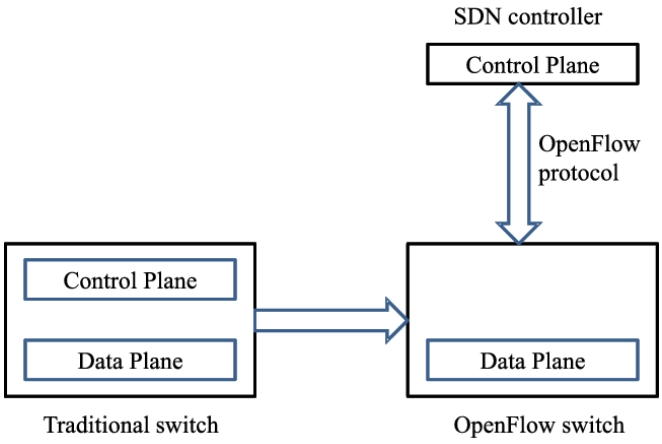

Fig. 2. Separation of data and control plan. [19]

two machines. Thus, we can identify more information about the attack, such as source and destination IP address, number of packets sent, thus having the perception of attack behavior in addition to equipment that may be infected with viruses, being part of a botnet [21], for the work was developed a honeynet with virtual machines, using the distribution Kali Linux.

C. Mininet and Open Virtual Switch

The Mininet and OVS software were used to build a testbed network environment, consisting of a linear topology with one switch and three client hosts. DoS attacks were generated from hosts using hping7 [22] and nmap tools. This work used several attacks directed to host IPs within the lab's internal network, within an isolated testing environment designed for this purpose.

Unlike the ping tool that only transmits Internet Control Message Protocol (ICMP) packets, hping3 transmits TCP, UDP, ICMP, RAW IP protocols and traceroute, identifying the path by which the packets went. Hping3 can be used to generate a large data stream for web servers, DNS or any application that operates on the TCP/IP stack, and is often used for penetration testing and vulnerability identification [23].

D. Communication between Open Virtual Switch and Controller

As defined by [24], the OpenvSwitch allows the creation of a virtual switch. This OVS implements protocols such as Spanning Tree, Link Aggregation Control Protocol (LACP), RSPAN, and can interact with real switches, building a hybrid network architecture. The switch configuration can be done via a command-line interface or from the OpenFlow protocol, which is the focus of this work. Before performing a packet forwarding, the switch asks the controller how it should be forwarded, setting up in its local memory a packet forwarding

table (flows). The next packets that match the rules described in the table are immediately forwarded by the switch, without the need for a new query to the controller, thus increasing forwarding performance. The switch can make a new query to the controller, checking if there is a new rule and if the current rules are still valid.

The OpenFlow-enabled switch counts the number of packets that matched a given rule, as shown in Fig. 3.

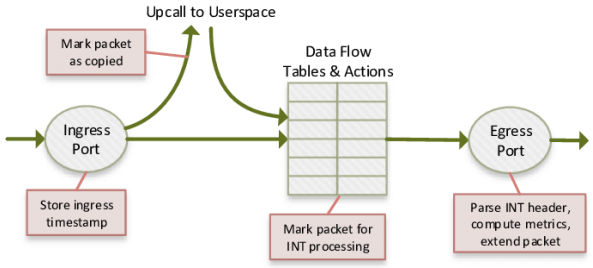

Fig. 3. OpenFlow Communication [25]

In this test, we set the interval for querying new rules every sixty seconds. This time was chosen to reduce the number of switch queries to the controller, but new rules could be added in a short time.

We have developed an API that, when identifying that anomalous traffic has passed through the IDS, adds a rule to the Ryu controller via the REST API, requesting traffic blocking.

V. Results

The architecture is validated, as described in Fig. 4. The network was built on Mininet, creating a virtual switch and a PC client. Virtual switch forwarding settings are configured on the Ryu controller. The virtual switch connects a virtual machine with Iptables and Snort receiving all packets directed to the Internet and DMZ and checking if they comply with any previously registered rule. Fig. 4 illustrates the beginning of an attack where Snort has not yet identified anomalous traffic generated by the LAN client. The hping3 tool installed on the PC initiates a denial of service attack intended for a DMZ IP. Until Snort detects the attack and requests the rule to be included in the controller, the destination server continues to receive all traffic generated by the attacker.

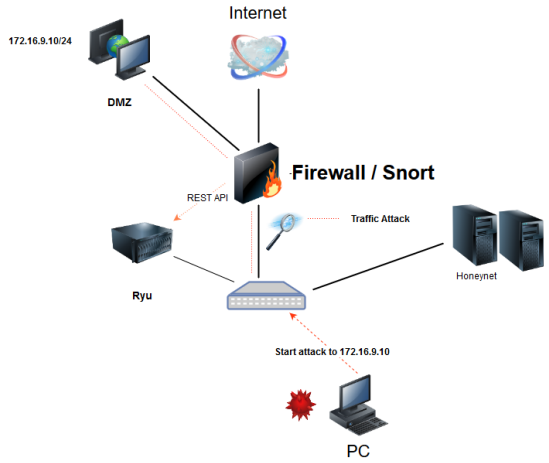

Fig. 4. Test scenario 1

We developed a python script that, based on events identified by Snort rules, makes a REST call to the Ryu controller, which adds the new rule to its flow table. The open Virtual Switch has been configured from mininet to query the controller every 60 seconds. The controller will receive from Snort a specific flow register, informing the source IP, destination, and destination port that the block should occur. This way, other non-anomalous data streams can pass as usual.

The default blocking rules are to direct traffic to honeypots, so that traffic will be logged by the server. The figure depicting the network after convergence is described in Fig. 5. When an anomalous event is detected, an email notification is sent to the administrator to take appropriate action. The rule remains in the controller for up to two hours and can be manually removed by the operator.

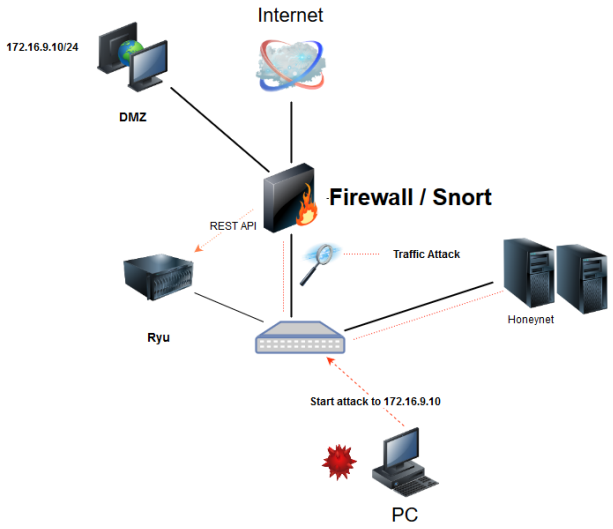

Fig. 5. Security Convergence SDN

To validate the concept, 100 tests, automated from python3 scripts. We performed the measure snort, identifies any unusual traffic; request the inclusion of a new flow in the table, and forces the switch to forward traffic to the honeynet. In

this case, To validate the concept, 100 tests, automated from python3 scripts, were performed to measure the time Snort identifies anomalous traffic, make the request to include a new flow in the table, and force the switch to forward traffic to the honeynet. In this case, the convergence time described in the Fig. is measured the convergence time described in the Fig. 4 and 5.

the convergence time, where the network protection occurs, obeys a linear function where:

$$C(x, y) = x + y \tag{1}$$

where x is the time it takes Snort to identify anomalous traffic and write the rule to the controller via API and y the time it takes the SDN network to converge.

Using hping3 with flood parameters enabled, directing traffic to port 80 of the server installed in the DMZ environment and running the tests 100 times identified the following x values presented in the Table I.

The time taken by the parent company to register new rules (y) was also measured in the tests and the results are presented in Table I.

TABLE I
RESULT OF LINEAR FUNCTION C(X,Y) = X + Y

	Snort-Time (X)	SDN Network-Time (Y)
Mean	3,47	30,44
Std. Desviation	1,71	17,60
3 Quantile	5	47

From the values presented in the table, we can state that the time for identification of an attack by Snort (x) suffers little variation in relation to the average. Not so with the time it takes for the switch to redirect traffic to the honeynet. This fact can be explained by the pooling interval between the switch and the controller set at 60 seconds. For example, if a new rule is registered one second after the switch has been queried, it will be necessary to wait 58 seconds for the new rule to be processed. The sample values are illustrated in the graph 6.

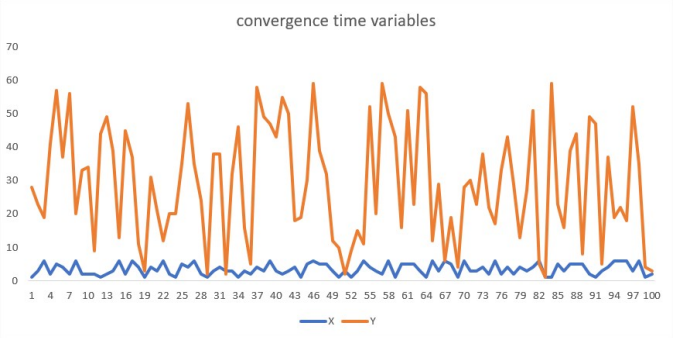

Fig. 6. Test results

VI. Conclusion and Future works

As presented in the results section, the proposed solution proved to be useful in identifying attacks and taking actions to mitigate DoS attacks from internal network devices. Convergence time is directly influenced by the pooling interval between the switch and the controller. Reducing this interval, however, increases the volume of traffic between the controller and the switch, delaying routing decisions. One of the future works is to develop a solution where the SDN switch is immediately warned of the inclusion of new rules, making network convergence faster. It will also include rules for targeting attacks from external networks, such as the Internet and DMZ.

References

[1] 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, Fourthquarter 2018.

[2] M. P. Fernandez, "Comparing openflow controller paradigms scalability: Reactive and proactive," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, March 2013, pp. 1009–1016.

[3] F. Y. Nagahama *et al.*, "Ipsflow: um framework para sistema de prevenção de intrusão baseado em redes definidas por software," 2013.

[4] B. Caswell and J. Beale, *Snort 2.1 intrusion detection*. Elsevier, 2004.

[5] T. Xing, D. Huang, L. Xu, C. Chung, and P. Khatkar, "Snortflow: A openflow-based intrusion prevention system in cloud environment," in *2013 Second GENI Research and Educational Experiment Workshop*, March 2013, pp. 89–92.

[6] D. M. F. Mattos and O. Duarte, "Qflow: Um sistema com garantia de isolamento e oferta de qualidade de serviço para redes virtualizadas," *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos-SBRC*, 2012.

[7] S. Kyung, W. Han, N. Tiwari, V. H. Dixit, L. Srinivas, Z. Zhao, A. Doupé, and G. Ahn, "Honeyproxy: Design and implementation of next-generation honeynet via sdn," in *2017 IEEE Conference on Communications and Network Security (CNS)*, Oct 2017, pp. 1–9.

[8] W. Fan, D. Fernández, and V. A. Villagrá, "Technology independent honeynet description language," in *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, Feb 2015, pp. 303–311.

[9] J. F. Kurose and K. W. Ross, "Redes de computadores e a internet," *Uma nova*, 2006.

[10] M. Yip, "Method and system of aggregate multiple vlans in a metropolitan area network," Jun. 28 2005, uS Patent 6,912,592.

[11] M. Gast, *802.11 wireless networks: the definitive guide*. " O'Reilly Media, Inc.", 2005.

[12] F. Andritsos, "Port security amp; access control: A systemic approach," in *IISA 2013*, July 2013, pp. 1–8.

[13] B. F. Cox, B. McMurdo, and V. R. Yarlagadda, "802.1 x authentication technique for shared media," Nov. 24 2009, uS Patent 7,624,431.

[14] C. Jeong, T. Ha, J. Narantuya, H. Lim, and J. Kim, "Scalable network intrusion detection on virtual sdn environment," in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Oct 2014, pp. 264–265.

[15] M. Monshizadeh, V. Khatri, and R. Kantola, "Detection as a service: An sdn application," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, Feb 2017, pp. 285–290.

[16] K. Nam and K. Kim, "A study on sdn security enhancement using open source ids/ips suricata," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct 2018, pp. 1124–1126.

[17] J. Kim and S. Shin, "Software-defined honeynet: Towards mitigating link flooding attacks," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, June 2017, pp. 99–100.

[18] M. A. Prado *et al.*, "Análise experimental da botnet iot mirai." 2018.

[19] M. Hamed, M. Fouda, B. ElHalawany, and A. S. Tageldien, "A novel approach for resource utilization and management in sdn," 12 2017.

[20] Ryu controller. [Online]. Available: https://osrg.github.io/ryu-book/en/Ryubook.pdf

[21] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.
[22] Hping7 nmap.
[23] B. Ops, "Denial-of-service attack–dos using hping3 with spoofed ip in kali linux," *BlackMORE Ops. BlackMORE Ops*, vol. 17, 2016.
[24] Open virtual switch reference. [Online]. Available: https://www.openvswitch.org
[25] A. Gulenko, M. Wallschläger, and O. Kao, "A practical implementation of in-band network telemetry in open vswitch," 10 2018, pp. 1–4.