Cross Security - A SDN based Firewall

Kabeer Ahmed, Haris Aqeel, Rehan Mumtaz, Muhammad Saad, Muhammad Areeb Nadeem

Abstract - Network Level attacks have been increasingly growing throughout the years. The safety of people and computers is now seriously threatened by this. The malicious actors utilize the TCP/IP stack in order to transmit malicious packets to the legitimate users and gain access to their private and personal information. In order to tackle this problem, we propose Cross Security - a firewall that uses SDN and is built on top of the POX controller. Programmable security policy control is made possible by decomposing the firewall architecture into its control and data planes. Using mininet emulator, we demonstrated the practical implementation of several hosts belonging to different networks and applied firewall rules on it. Experimental results show that using our approach, we successfully filtered the desired packets in and out of the networks.

Keywords - Software Defined Networks, Mininet, POX Controller, Firewall, TCP/IP stack

I. Introduction

NETWORK administration with software-defined networking technology is more similar to cloud computing than traditional network management since it allows for dynamic, programmatically effective network setup to enhance network performance and monitoring. Modern applications' high-bandwidth, dynamic nature makes SDN, an emerging architecture, the ideal fit. SDN is flexible, dynamic, inexpensive, and controlled. The network control may be directly programmed and the underlying infrastructure can be abstracted for use by applications and network services since the network control and forwarding activities are separated in this configuration. Software-Defined Networking (SDN) enables various network applications, which are effectively network services, to operate on the controller and directly manage the network by specifying packet-handling mechanisms in underlying devices [1]. First, SDN separates the devices that make up the data plane from the control plane, which decides how to manage the traffic and sends it in line with those decisions. Due to the separation of the control and data planes, the network's underlying devices are reduced to straightforward dump forwarding devices, and the control logic is implemented in a logically centralised controller that acts as the NOS, creating a consistent and up-to-date network picture. Thirdly, SDN enables a single software application to simultaneously operate many infrastructure layer devices by strengthening the empowerment of the control plane. One advantage of decoupling the control plane and data plane is the simplicity with which a distributed set of underlying network devices can be managed and designed. Another advantage is an abstraction paradigm that facilitates autonomous evolution of the control and data planes [2]. An example of a network security system is a firewall, which monitors and controls incoming and outgoing network traffic in accordance with preset security rules. Using an SDN concept, it is possible to manage and control network devices in a flexible manner. It enables the installation of firewall policies on edge devices and the remaining switching network hardware. The deployment of a firewall close to a network device that produces forbidden traffic improves network performance. This lessens the amount of unnecessary packets that are sent across the network [3]. The most important component of SDN is a communication protocol between the control plane and the data plane. The Open Networking Foundation, a global SDN organisation, has supported the OpenFlow Protocol in order to achieve that goal (ONF). The OpenFlow switches feature flexible packet processing since the controller is now being asked how to handle the incoming data [4]. Several public SDN controllers, including Floodlight, Ryu, NOX, and POX, now support OpenFlow.

The rest of the paper is organized as: In Section II the Related Work related to this research is discussed. Section III describes the methodology, tools techniques utilized while implementing the firewall. Section IV discusses the experimental setup, Section V is results and analysis of our approach. Section VI concludes the paper and the research work.

II. Related Work

The difficulty of recognising horizontal port scans on home networks is the focus of the study in [5]. They suggest a firewall technology based on software-defined networking (SDN) that can recognise horizontal port scans. Current SDN implementations (like OpenFlow) do not provide access to packet-level data, which is required for network security applications because of speed limitations. To access packet-level data, their architecture makes use of FleXight, a novel information channel they suggest between an SDN controller and data route components. They put the technique to the test using a big actual packet trace from an ISP, showing that our system can identify every attacker and 95% of vulnerable targets with just 0.75 percent network overhead.

In this study [6], the stateful firewall was added to the data plane of the SDN switch. It utilizes the Open vSwitch. They assess the SDN stateful firewall's performance as well. The findings demonstrate that our SDN stateful firewall can operate properly with only a little increase in overhead in SDN switches.

The formal modelling that utilises the pACSR process algebra is first discussed by the authors of this [7] study, who then offer a method for validating the firewall application

that runs on top of the SDN controller. The firewall rules are translated into a pACSR process, which serves as the specification, and the packet behaviours in SDN are translated into a pACSR process, which acts as the implementation. They then check to see if the parallel composition of two pACSR processes is free of deadlocks to show the accuracy. Additionally, we can instantly apply our verification to the case of changing the network architecture to determine whether any anomalies or inconsistencies would appear.

Internet service providers (ISPs) are said to provide a firewall service that enables end users to request and apply match-action rules in ISP edge routers, according to a study [8]. In the fictitious scenario, an ISP separates the control plane from the data plane in an SDN environment using OpenFlow and an ONOS controller. In order to facilitate communication between users and the SDN Controller through a secure SSL/TLS connection, the author provides an application programming interface (API). The Controller must convert high-level logic into low-level rules for OpenFlow switches.

ChainGuard [9] implements a firewall for blockchain applications by using SDN features to filter network traffic in order to offer additional security to the blockchain nodes. To establish whether the origin of the traffic is valid, ChainGuard talks with the blockchain nodes it is charged with protecting. Unauthorized packets are captured and are therefore unable to affect the blockchain.

This study [10] provides a selective firewall distribution method that is topology aware and transmits only the firewall configuration rules that are really required given the traffic patterns and network topology. The Mininet simulation results for various network sizes demonstrate that the proposed method considerably decreases firewall setup traffic and the travel path for traffic that violates the firewall, making it appropriate for large-scale SDN networks.

This article [11] focuses on the virtual networking domain's replacement of physical switches as well as the creation of firewall and load balancing applications that run on OpenFlow-based SDN controllers. Additionally, it shows that, because software can be used to create a firewall, the majority of dedicated hardware firewall devices may be replaced.

The researchers in [12] present ACLSwitch, a virtual firewall that covers the entire network and use the OpenFlow protocol to filter traffic in a network of OpenFlow switches. They developed "policy domains" that enabled different network switches to have different filtering settings applied to them. The method enables rules to be distributed throughout a network without needing a human operator to individually transmit the rules to switches.

In this study, they offer a firewall solution for SDN that is based on MAC filtering [13]. The performance of the average packet delay for the firewall techniques using MAC filtering and IP filtering is then compared. The results show that while the security support offered by both types of firewall strategies (packet allow/deny) is nearly similar, the MAC filtering firewall outperforms the IP filtering firewall in the delay performance research.

In order to protect network-attached devices from malicious and suspicious traffic about attacks, this article [14] introduces firewalls for SDN infrastructure. Access control and distribution systems are used to improve SDN security.

They [15] developed a rigorous procedure for flow space and firewall authorization space checking in order to identify and resolve problems in OpenFlow-based firewalls. Their approach may investigate conflicts between firewall rules and flow policies based on all flow channels inside an OpenFlow network. They also introduced the testing of intra-table dependencies for flow tables and firewall rules.

In this work [16], a layer 3 firewall solution employing a complete mesh topology with a single controller, six switches, and a single host on each switch is shown. Using the Mininet network emulator, the Learning switch's POX controller code is modified for a complete mesh topology, and the packet flow between the hosts is regulated in accordance with the rules introduced into the Learning switch using the OpenFlow controller.

This paper [17] examines the difficulties in implementing stateful firewalls in an SDN environment and introduces FlowTracker, a novel stateful firewall solution that aims to preserve stateful firewall accuracy and agility while reducing controller processing and communication overhead between the control and data plane.

In this study, they look into whether SDN may enhance network security [18]. In particular, the idea of combining multiple SDN capabilities with security functions is thoroughly studied. To demonstrate the viability of SDN-based security functions, they implement four different types in the Floodlight applications: in-line mode security functions (like firewalls and IPS), passive mode security functions (like IDS), network anomaly detection functions (like scan and DDoS detector), and advanced security functions (e.g. stateful firewall and reflector networks).

In a cloud-clustered context, this work [19] proposes a reliable technique for distributing firewall security policies to dispersed SDN devices.

This article [20] focuses on building and creating firewall applications based on OpenFlow. The execution demonstrates that the majority of firewall features may be developed using software, without the requirement for specialized hardware. For our studies, they employed the free source POX Controller, which is based on Python. They installed Mininet emulator and utilized VMPlayer virtualization software to create network topologies for the experiment.

III. Methodology

Flow control is offered by a programme known as a software-defined networking (SDN) controller, which improves network management and application performance. Switches are instructed where to transfer packets by the SDN controller platform, which typically operates on a server [21]. We first set up the firewall using an SDN Controller. The controller is programmed in such a way that it blocks or forwards certain TCP/UDP transmissions. Each TCP/IP Transmission can be uniquely identified by the following properties, i.e. Source IP, Destination IP, Source Port, Destination Port and the Transmission Protocol.

The SDN Controller uses OpenFlow, a communications protocol, to control the network by granting access to a switch or router forwarding plane via a network. An OpenFlow switch, a piece of hardware or software that forwards packets in software-defined networking, is utilized in order to create the firewall.

The entire SDN Firewall works in such a way that every time the OpenFlow switch receives a packet that begins a new flow, it sends it on to the SDN controller, who then determines whether to advance the packet to the other network or to stop it entirely. This choice is dependent on the firewall's configured rule set and current working mode (pass-through, white-list, or black-list). When the controller makes such a choice, it installs the proper forwarding rule in the switch to ensure that subsequent packets belonging to the same flow are treated in the same way.

As was previously noted, the firewall serves as an SDN controller. It is made with the POX framework and the Python programming language. It provides an API that enables adjusting the firewall's functionality (for example, adding and removing forwarding rules) and altering the work mode of the firewall.

In order to properly visualize the working of the firewall, a Web based UI is created using python's Flask framework. It offers an API that may be used to change the firewall's settings, such as its current working mode or to add or remove forwarding rules, among other things.It also permits looking for certain statistical and event-based information about the traffic that went through the firewall.

IV. Experimental Setup

We prototyped our network using a mininet emulator in our actual experiment. Mininet allows users to quickly create a realistic virtual network running real kernel, switch, and software application code on a personal computer and to mimic a network architecture using Openflow switches. Mininet also allows users to easily design, interact with, personalise, and share an SDN prototype [22]. We created our network topology using Mininet as can be seen from the fig. 1. In our topology, we created 1 SDN Controller, 3 OpenFlow Switches (s101, s102, s200) and 4 hosts (h1, h2, h3, h4).

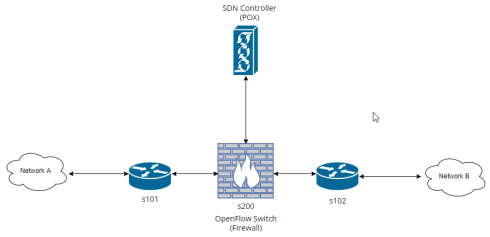

Fig. 1.

Network Architecture

The OpenFlow switch s200 behaves like a firewall by connecting the switch with our SDN Controller (POX Controller). For software-defined networking (SDN) control applications like OpenFlow SDN controllers, POX is an open source platform built on Python. The 4 hosts are divided into two pair or networks i.e. Network A Network B. If a transmission is generated from Network A (h1 and h2) towards Network B (h3 and h4) then the transmission first hits the firewall. The firewall then filters the flow using the specified set of rules according to the selected Mode i.e. White-List, Black-List and Pass-Through and then if the configuration allows, forwards the transmission to the other network and block otherwise.

V. Results & Analysis

To analyze and test the firewall's performance, we tested the application using different scenarios. At first, we configured the firewall in such a way that it blocks all the traffic coming from host h3 with IP Address 10.0.0.3 to host h1 with IP Address 10.0.0.1 on Destination port 80. The results can be seen from fig 2. That the firewall blocked the traffic as per the configuration.

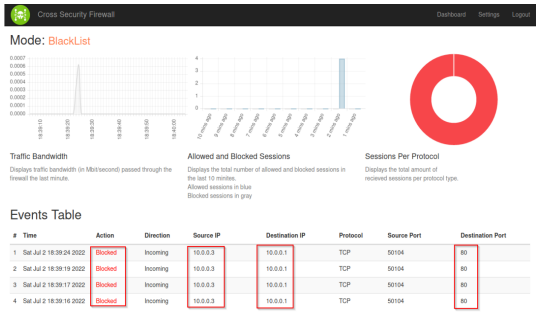

Fig. 2.

Blocked Sessions

Similarly we can cross check the scenario by accessing the web server (port 80) on host h2 with IP Address 10.0.0.2 from the host h3 with IP Address 10.0.0.3. As can be seen from fig. 3. The application Allowed the traffic when the source was h3.

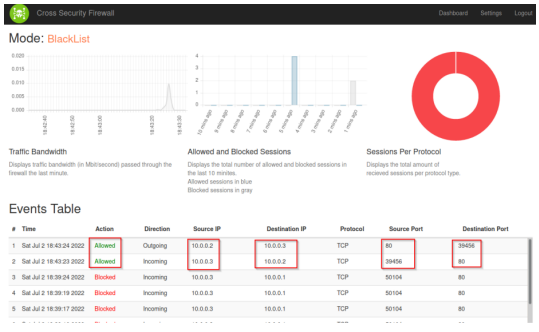

Fig. 3.

Allowed Sessions

We can also add new rules for the upcoming traffic and transmissions on the selected mode easily by the Web UI as shown in fig. 4. On the main dashboard, we can also see the real time bandwidth, Allowed and blocked sessions in the last 10 minutes and protocol information.

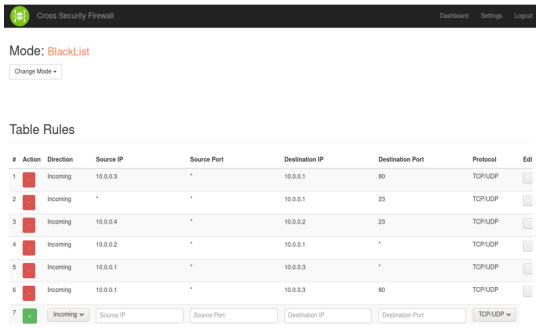

Fig. 4.

Add/Remove Firewall Rules

VI. Conclusion

Firewalls prevent the flow of malicious traffic and help in ensuring the security of the network. In this paper, we implemented a SDN based firewall. The purpose of this firewall is to block and allow relevant traffic in accordance with the configuration of the firewall. Using the mininet emulator, we successfully demonstrated the implementation of different switches which are connected to different hosts. The firewall filters the traffic using the POX controller and Networks A and B are separated on the specified rule set. Experimental results and analysis show that our proposed solution successfully blocks the desired transmissions according to the given mode.

References

[1] Hongxin Hu, Gail-Joon Ahn, Wonkyu Han, and Ziming Zhao. Towards a reliable {SDN} firewall. In *Open Networking Summit 2014 (ONS 2014)*, 2014.

[2] Wajdy M Othman, Hao Chen, Ammar Al-Moalmi, and Ali N Hadi. Implementation and performance analysis of sdn firewall on pox controller. In *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, pages 1461–1466. IEEE, 2017.

[3] Sergey Morzhov, Igor Alekseev, and Mikhail Nikitinskiy. Firewall application for floodlight sdn controller. In *2016 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–5. IEEE, 2016.

[4] Michelle Suh, Sae Hyong Park, Byungjoon Lee, and Sunhee Yang. Building firewall over the software-defined network controller. In *16th International Conference on Advanced Communication Technology*, pages 744–748. IEEE, 2014.

[5] Sajad Shirali-Shahreza and Yashar Ganjali. Protecting home user devices with an sdn-based firewall. *IEEE Transactions on Consumer Electronics*, 64(1):92–100, 2018.

[6] Pakapol Krongbaramee and Yuthapong Somchit. Implementation of sdn stateful firewall on data plane using open vswitch. In *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–5. IEEE, 2018.

[7] Miyoung Kang, Jin-Young Choi, Inhye Kang, Hee Hwan Kwak, So Jin Ahn, and Myung-Ki Shin. A verification method of sdn firewall applications. *IEICE Transactions on Communications*, 99(7):1408–1415, 2016.

[8] Andis Arins. Firewall as a service in sdn openflow network. In *2015 IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, pages 1–5. IEEE, 2015.

[9] Mathis Steichen, Stefan Hommes, and Radu State. Chainguard—a firewall for blockchain applications using sdn with openflow. In *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, pages 1–8. IEEE, 2017.

[10] Thuy Vinh Tran and Heejune Ahn. A network topology-aware selectively distributed firewall control in sdn. In *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 89–94. IEEE, 2015.

[11] Nayana Zope, Sanjay Pawar, and Zia Saquib. Firewall and load balancing as an application of sdn. In *2016 Conference on advances in signal processing (CASP)*, pages 354–359. IEEE, 2016.

[12] Jarrod N Bakker, Ian Welch, and Winston KG Seah. Network-wide virtual firewall using sdn/openflow. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 62–68. IEEE, 2016.

[13] Perumalraja Rengaraju, S Senthil Kumar, and Chung-Horng Lung. Investigation of security and qos on sdn firewall using mac filtering. In *2017 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5. IEEE, 2017.

[14] Dhaval Satasiya, Rupal Raviya, and Hiresh Kumar. Enhanced sdn security using firewall in a distributed scenario. In *2016 international conference on advanced communication control and computing technologies (ICACCCT)*, pages 588–592. IEEE, 2016.

[15] Juan Wang, Yong Wang, Hongxin Hu, Qingxin Sun, He Shi, and Longjie Zeng. Towards a security-enhanced firewall application for openflow networks. In *International Symposium on Cyberspace Safety and Security*, pages 92–103. Springer, 2013.

[16] Avinash Kumar and NK Srinath. Implementing a firewall functionality for mesh networks using sdn controller. In *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pages 168–173. IEEE, 2016.

[17] Thuy Vinh Tran and Heejune Ahn. Flowtracker: A sdn stateful firewall solution with adaptive connection tracking and minimized controller processing. In *2016 International Conference on Software Networking (ICSN)*, pages 1–5. IEEE, 2016.

[18] Changhoon Yoon, Taejune Park, Seungsoo Lee, Heedo Kang, Seungwon Shin, and Zonghua Zhang. Enabling security functions with sdn: A feasibility study. *Computer Networks*, 85:19–35, 2015.

[19] Yuwei Chang and Tsungnan Lin. Cloud-clustered firewall with distributed sdn devices. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–5. IEEE, 2018.

[20] Karamjeet Kaur, Krishan Kumar, Japinder Singh, and Navtej Singh Ghumman. Programmable firewall using software defined networking. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2125–2129. IEEE, 2015.

[21] Jennifer English. What is sdn controller (software-defined networking controller)? - definition from whatis.com, Dec 2018.

[22] TechTarget Contributor. What is mininet? - definition from whatis.com, Mar 2013.

BIOGRAPHY

Kabeer Ahmed is an undergraduate student at NED University of Engineering and Technology in Software Engineering. His main area of interest are Offensive Security like OS Exploitation, Web Exploitation and Defensive Security like Digital Forensics, Cryptography. He has hands on experience in Data Extraction and web penetration testing. He is keen interested to make this career in this Cyber Security field.

Haris Aqeel is a student of NED University, currently doing B.E in Software Engineering. His main area of interest are Blockchain and web development. He has experience of different web application development from requirement gathering phase till deployment.

Rehan Mumtaz, a software undergraduate at NED University holds a strong interest in the chain of development and breaking of the whole loop set of technologies. I am a great fan of cyber-security and loves to research about this domain and takes this as my passion as my expertise solely lies in Penetration testing(especially web and operating system exploitation), Cryptography Exploit development. I am looking forward to move ahead with my passion and progress with it in the future InshaAllah

Muhamad Saad is a Student of Software Engineering at NED University of Enineering Technology. He is passionate and enthusiast in the field of Cyber Security. His areas of expertise are Penetration Testing, Malware Analysis Reverse Engineering. Saad has hands on experience in Web Network Penetration testing. His future aim is to make a career in this domain.

Muhammad Areeb Nadeem is an undergraduate student in the Software Enginerring Program at the Ned University of Enginerring and Technology.His main areas of interst are data analytics,cyber-security and web development.He has experienced of data cleaning ,data modeling,data analytics tool like powe bi.He has also hand on experience in cyber security tools like wireshark,nmap,burpsuite etc.His future aim is to become a data analytical engineer and purse his carrier in the field of data science.