Implementation of Stateful SDN Firewall

Madusha Jayawardhana, Ninada Perera, Priyankara Bandara, Suneth Namal and Sampath Deegalla
Department of Computer Engineering
University of Peradeniya, Sri Lanka
{e11487, e11297, e11040}@gmail.com, {namal, dsdeegalla}@pdn.ac.lk

***Abstract*— SDN Firewalls have become very prominent in modern networking security. Since Software Defined Networks separate control logic from network devices and act as a centralize controller, most of the traditional networking issues could be solved. Thus major disadvantages of traditional firewalls are overcome by SDN Firewalls. Moreover, SDN firewalls are more advanced in packet handling.**

Objective of this project is to implement a stateful firewall that runs on the Floodlight runtime. We develop a distributed flow-based firewall prototype and test it on a simulated network through Mininet. Also a data mining model was integrated to the firewall to automate the decision making process to identify network intrusion detection.

Keywords—SDN, Firewall, Data Mining, Network Security

I. INTRODUCTION

SDN service chaining uses software to insert services virtually into the flow of network traffic. A centralized controller can connect multiple network and security services in a series-or chain-across network devices. With SDN service chaining, networks can be reconfigured on the fly, allowing them to dynamically respond to the needs of the business. For both enterprises and service providers, this means that SDN service chaining will dramatically reduce the time, cost, and risk for them to design, test, and deliver new network and security services. Security service chaining for SDN has many benefits, such as the ability to elastically scale stateful firewalls that run as virtual machines. It is all based on need and dynamically adjustable as instances of services come and go. Scrub traffic, protecting various elements of the SDN, authenticate and ensure authorization of changes to the dynamic network, optimizing traffic flows, correlate network changes with security changes and Creating and associating new service classes are couple of requirements the uses may come across in daily basis.

The progression of secure SDN will entail taking out the stateless Access Control Lists (ACLs) and providing better protection through stateful inspection and all the other goodies that come with purpose-built virtualization security—including compliance, introspection, intrusion detection, etc. It will make for a more solid solution by ensuring that no one can disable the controller.

If every Ethernet switch in the environment could perform like a traditional firewall, it would change the way security policy is implemented in a networked environment. Assuming that every Ethernet switch was a multi-port firewall, and then firewall policies could be implemented throughout the network at every ingress switch port and on every link between switches. There would be firewalls for every server, desktop, every link. The policies for these firewalls would be implemented by a controller that maintains the global view of the current application traffic. Having security policies enforced throughout the environment would mean the complete erosion of the security perimeters. Having that many security policies implemented and maintained manually would be an administrative nightmare. However, with centralized controller architecture, the policy would be created once and then pushed down to every network device for enforcement.

The key concept to the feasibility of using an SDN-enabled switch as a firewall is the state that it would maintain of the application traffic flows. ACLs are not stateful and do not have awareness of when the connection started or ended. However, SDN switches as firewalls allow to create stateful firewalls. A firewall does not handle packets, but the flows. That is, the flow between a client and a server from start to the end of a Session. The firewall need to handle all flows in a session with stateful packet inspection. This allows to take the control over the packets being transmitted. Thereby the network will be able to self-manage the resources once traffic is configured at the start.

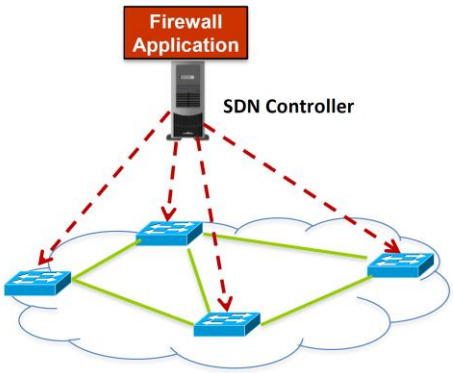

Figure 1.1. SDN firewall architecture

II. METHODOLOGY

The solution was to provide better security for the networks through SDN firewall by using stateful inspection and all the other good features that come with purpose-built virtualization security and thereby try to automate the process of SDN firewall decision making, done at the SDN controller using data mining.

The proposed solution was focusing on implementing a firewall module and run it on the SDN controller of the simulated network and thereby collect a set of sample data from the firewall information logs for further studies. The training data set obtained was to be used in data mining process in order to come up with an algorithm to automate the process of firewall configuration at the SDN controller.

First the stateful firewall should be implemented. There are several SDN controllers available such as POX, NOX, Beacon, Floodlight, Ryu, Trema and etc. NOX is the first SDN controller and POX which is an extended version of NOX is a python based controller[1]. The firewall application is coded with the same a programming language used for coding the controller. Floodlight[2] is a Java based SDN controller with built in stateless firewall application. Available SDN firewalls were only supporting rule based stateless firewall implementation. Our design was to create a stateful firewall on SDN controller. Figure 2.1 shows the overall architecture of the solution.

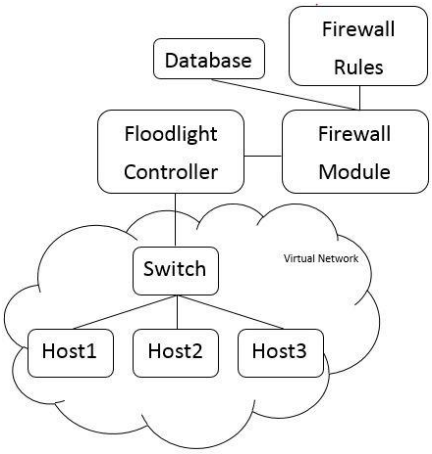

Figure 2.1 Overall architecture of the solution

The network which consists of switches running Openflow protocol is controlled by a centralized SDN controller. The stateful firewall module implemented, is residing in SDN controller, providing security for the network. SDN testbed was created using Mininet which was a system that provided scalable, virtual software defined networks for rapid prototyping on a single computer. OpenFlow was used as the standardized SDN protocol which defined a way for the controller to communicate with switches. OpenFlow provides the capabilities of centralized control of multiple switches from a single controller, dynamic forwarding information update and analysis of traffic statistics at switches [3].

FLoodlight controller was selected as the SDN controller due to its recognition for the features and performance. It is a java based controller which supports the Openflow versions from 1.0 to 1.4. Floodlight is considered as a robust controller which is easy to use. These were some reasons behind choosing Floodlight as the SDN controller.

Finally a simulated network is built on mininet network simulator and random network traffic is generated from hosts to the servers. Then some attributes of all the connections were stored to a database. Connection details stored on the database were then used to train & test a data mining model which is integrated to the firewall later. After the integration the firewall is able to identify any suspicious activity & alert the concerned parties.

III. Implementation

Before building the firewall, a virtual network should be simulated. Mininet is used to simulate the virtual network and Floodlight is used to create the controller. Then the firewall module can be integrated in to the firewall Foodlight controller. Next different network topologies were created using mininet. An example is given below. The network was created with 81 hosts & 40 OpenFlow switches connecting in tree topology which had a depth of 4 & branching factor of 3. And 5 of the nodes were configured as http servers.

```
sudo mn --topo tree,depth=4,fanout=3 --
controller=remote,ip=192.168.246.128,port=6653
--switch=ovsk,protocols=OpenFlow13
```

In basic packet filtering the firewalls does not keep track of the state of the ongoing connection sessions. This may easily lead to DoS attacks where external users can flood internal hosts with malicious packets. These firewalls are called stateless firewalls. But stateful firewalls address the problems of stateless firewalls by keeping track of ongoing connection sessions. Stateful TCP packet filtering is easier due to the presence of TCP flags (SYN, ACK, FIN, PSH, URG, and FIN). But tracking of UDP session state is a complicated process[4]. Because UDP is a connectionless transport protocol which does not contain any sequence numbers or flags like TCP. Therefore, we planned to implement the stateful firewall only for TCP, in the limited time frame we had.

The existing rule based stateless firewall module of Floodlight was used and modified according to the need. The firewall enforces Firewall rules which are stored in a text file in comma separated format. For example, the rules stored in the following format as source IP address, destination IP address, protocol allows the packets to pass through the network.

10.0.0.1,10.0.0.4,ARP, ALLOW

10.0.0.4,10.0.0.1,ARP, ALLOW

The connection state of ongoing TCP connections was stored in a hash map in the memory. A combination of source and destination IP addresses together with their TCP ports were used to uniquely identify a connection flow. TCP Flags in TCP packet headers were checked in order to determine the state of these connections.

The firewall inserts a new connection flow into the Hash map and sets its state to SYN_SENT, when a SYN packet is received. And once the two other remaining packets of the three-way handshake process are received, the TCP connection state transits to the ESTABLISHED state. Therefore, that flow can easily travel through the firewall. When the TCP connection is terminated with the reception of FYN packets, the firewall removes the connection flow entry from the hash map & blocks the connection.

The flow chart represented in figure 3.1 shows the flow of work in stateful firewall module. Incoming packets to the switches are sent to the controller if there is no matching flow in the switch. Packets sent to the controller are checked against the predefined firewall rules in the controller. Packets which are not allowed are dropped and packets which are allowed are further processed. Allowed ARP, UDP and ICMP packets are forwarded through the network. Allowed TCP packets are examined in order to maintain the state of the connections. Then the packets are allowed to pass through the network. But the packets which try to violate the connection flows are dropped. A retransmission counter is used to prevent syn flood attacks. If syn packets belonging to a certain connection flow are transmitted more than a threshold value, those packets are blocked. Furthermore, TCP packet transmissions among hosts without establishing a TCP connection are also prevented.

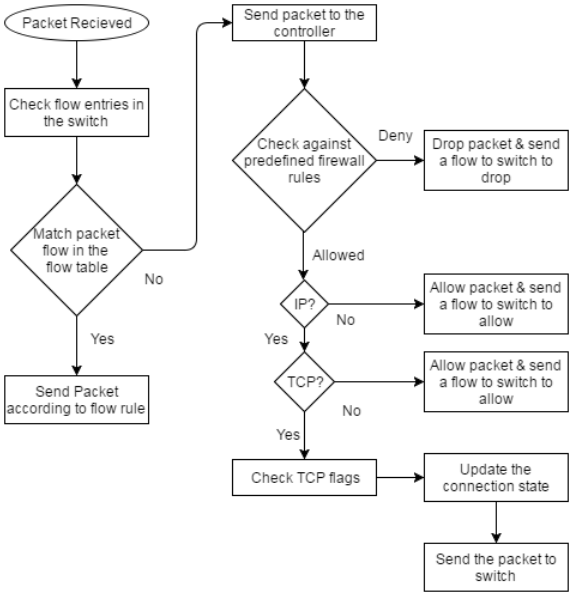

Figure 3.1. Flow chart

Random TCP traffic was generated from the hosts to servers using IPERF tool with different configurations. Then important details about every connection, were stored in a database for further studies. Information like connection duration, source ip, source port, destination ip, destination port, no. of bytes transferred in a connection, connection flag, number of packets of a connection, number of retransmissions of syn packets were stored. This was basically done for the future use, to obtain a data set in order to apply data mining techniques.

IV. Results And Discussion

Stateful firewall module that maintains connection states of TCP connections was built on floodlight controller. Once the connections were completed the details about connections was stored in a database for future use.

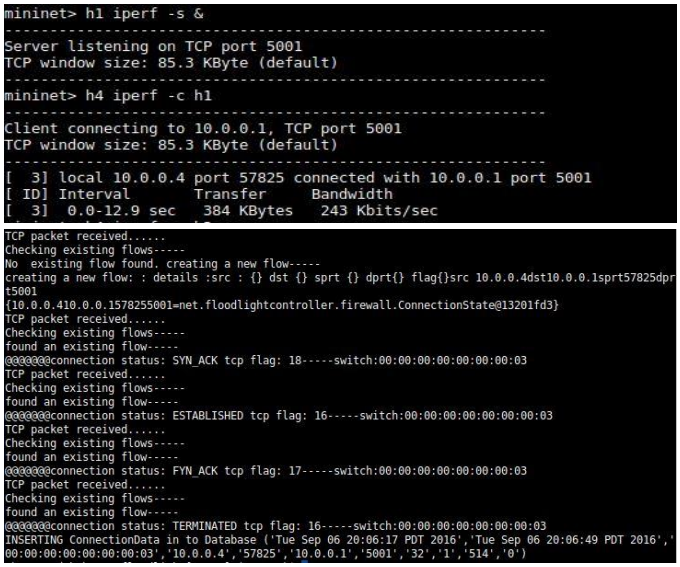

Figure 4.1. connection state maintained for a successful TCP connection

The connection state maintained for a TCP connection between host1 and host4 in virtual mininet network is shown in figure 4.1. Host1 was configured as TCP sever and host4 as TCP client. Connection state of the connection changes with the TCP flags exchanged between hosts. And once the connection is terminated details of the connection was stored in to the database. Packet blocking using SYN retransmission counter feature was not be able to test using the virtual network environment due the inability of simulating that network behaviour.

As further improvements the data collected using the firewall can be used in data mining and thereby come up with a data mining model to detect unusual network behaviour that can occur in the network. With these improvements we can automate the security process at the SDN controller.

V. Conclusion

SDN firewalls have become a prominent role in modern day security. By integrating stateful features to an SDN firewall makes the firewall more intelligent. The Stateful SDN firewall that we created by maintaining the state of the TCP connections prevents SYN flood attacks and DoS attacks that come from impersonated attackers. The controller has been made intelligent to analyze the network behavior and act more like distributed firewall. SDN is still evolving in so many areas and further research should be done to optimize the network security by covering all the aspects.

References

[1] Shieha, Alaauddin, "Application layer firewall using Openflow" (2014). *Interdisciplinary Telecommunications Graduate Theses & Dissertations.* Paper 1. http://scholar.colorado.edu/tlen_gradetds/1

[2] Floodlight: Open SDN Controller. [Online]. Available: http://www.projectfloodlight.org

[3] Raphael Eweka, Sangeetha Elango. "Implementation of address learning/packet forwarding, firewall and load balancing in Floodlight controller for sdn network management."

[4] Zouheir Trabelsi. "Teaching stateless and stateful firewall packet filtering: a hands-on approach"