Security Framework for Software Defined Networking with Openflow

Vaibhav K Matalia¹ Girish Khilari² Rahul Ekbote³
¹Senior Consultant ²Vice President (Persistent Systems) ³GTU PG School, Gandhinagar
¹Department of Information & Technology Systems and Network Security ²DAC, Pune

Abstract—The software defined networking is an emerging three layer architecture which defines control, data and application plane. Control and data plane implement routing and forwarding function, respectively. Application plane include Communication process. There are several firewall application built around controller for detect malicious traffic. At controller side some time traffic congestion is occurs so it hard to detect traffic according to rules. This study focuses on developing a firewall at switch level that protects our internal network. It reduces traffic congestion because malicious traffic is detected at switch level rather than controller side so it will provide more security. Our Implementation uses tree topology with one controller, three switches and four hosts. We have chosen POX controller written in python for the experiment and for creating SDN topology we use Mininet.

Keywords: SDN, Openflow, POX, Mininet, Firewall

I. INTRODUCTION

Software-defined networking (SDN) [1] is an emerging networking paradigm that gives hope to change the limitations of current network infrastructures. It separates data and control planes bundled inside a vendor closed router. First, it separates the network’s control logic from the underlying switches and router that forward the traffic. Second, with the separation of the control and data planes, network switches become simple forwarding devices and the control logic is implemented in a logically centralized controller. A simplified view of this architecture is shown in Fig.1. Controllers have a direct control over the data plane elements via Southbound Application programming interface as shown in Fig.1. The benefits of this approach are centralized configuration of network devices across organization computer network setup, less network traffic, reduced computation at router and ease of experiment. Replacement of network device in any organization is cumbersome task. With SDN architecture, replacement or upgrade are easy as configuration settings are applied from a central place.

SDN include two important milestone are Openflow and POX. Openflow include Openflow switch specification shown in Fig [2]. Openflow switch contain secure channel, and Openflow protocol. It also contains one or more flow table. Each flow table contains packet handling rules. Rules are match against traffic and perform certain actions like (dropping, forwarding, modifying etc) on traffic. POX controller is an SDN controller written in python. Component of POX based network are: Openflow Switches, controller process and database, see fig [3]. Openflow switch behave like firewall, it allows or reject specific type of information. This paper present firewall at switch level which protects internal network by modifying code provided with POX controller.

II. SECURITY ANALYSIS OF SDN

The basic properties of a secure communication in SDN are: Confidentiality, integrity, availability, authentication and non repudiation [4]. In order to provide a network protected from malicious attack or damage, network admin must secure the control plane, data plane and communication transactions across the network. When considering SDN, We need to focus on security aspect of seperate control and forwarding framework.

Much research has been done on examining security at control plane. For this purpose Firewall and IDS application is built around controller. Different application display different characteristics that make them more or less suitable to run in controller. But this Firewall and IDS application which is built around controller is suffered from traffic congestion at controller side. Because every time when packet arrives at switch it will go to controller for traffic checking. Controllers decide whether packet is malicious or legitimate.

Openflow switch specification [2] use transport layer security for communication between controller and switch. In this Openflow specification security feature is optional, and the standard of TLS is not specified. Because of this lack of vulnerability in TLS attacker is able to insert fraudulent rule, rule modification and possibility of DoS attack is increase. Due to centralized controller and
programmability of the network, new threat is introduced. For detecting this threat some techniques is propose like replication, diversity and secure component.

III. BACKGROUND

A. Openflow Switch

An Openflow Switch includes one or more flow table and a group table. They check packet lookups and forwarding. Switch and controller communicates with each other via Openflow protocol. With use of Openflow protocol controller manages the switch and controller is also able to add, delete, update flow entries in flow table. Flow table contain flow entries, each entry contain match field, counter and instruction to apply to matching packet. Each Flow entry include four basic action these are (1) forwarding a certain flow's packet to a given port, (2) encapsulation and forwarding the flow's packet to a controller, (3) dropping the flow's packet and lastly (4) forward the flow's packets through the switch's normal processing pipeline [3]. Initially Matching start at first flow table and may continue to additional flow table. Packet matching is done according to priority order. If a matching entry is found, the instructions associated with the specific flow entry are executed. If not then output depends on configuration of the table-miss flow entry. There are two types of Openflow switch, Openflow only and Openflow hybrid switch.

B. SDN Terminology

To recognize the different elements of an SDN as uniquely, now present the essential terminology.

1) Forwarding devices:- These are the hardware or software based data plane device that perform set of elementary operation. The forwarding devices have well-defined flow rules used to take decision against incoming packet. Flow
2) Rules are installed in the forwarding device by the SDN controller implementing the southbound protocols.
3) Data plane:- Forwarding device are interconnected through wireless channels or wired cables, which present the data plane. So it is a set of router, switches and bridges allow packet moves from point A to B.
4) Control plane:- Forwarding devices are programmed by control plane. The control plane is seen as "network brain".
5) Southbound API:- It allows switches to communicate with controller. With help of open flow protocol switches and controller can communicate with each other.
6) Northbound API:- It presents a programmable API to network. It supports all the services and application running over the network.

C. Firewall Design

The necessary function of firewall is to provide security to a network. Firewall authorizes or deny packet depending on the rule set applied. Firewall may perform an operation at Data-link layer of OSI. At this level the only basis for the firewall is the packet header information for determining whether a certain packet comes from a trusted source or not [3]. Filtering rules can be based on source and destination MAC address, port number. This concept is for simple packet filter firewall. This study will focus on the firewall which will protect our internal network. One basic algorithm suggests by author who involves waiting for incoming traffic, reading the packet header information check the rules list and finally decide whether packet is legitimate or not. In Openflow standard, the user interacts with entire network via controller. Firewall implementation is built at controller, so controller filters the traffic according to rules. However, an implementation like this would not yet take full advantage of Openflow. In order to take advantage of OpenFlow, the rule is installed in Switch instead of controller. Thus Switch may behave like Firewall, so the packets would not have to go to the controller each time for processing.

IV. EXPERIMENTAL setup

Our experimental setup based on study[5] which used Mininet IT provides virtual network as shown in Fig.3. It creates virtual connection, host processes and network name spaces. We used VMWare for virtualization. Topology is added in Mininet virtual machine as shown in Fig.4. Virtual network is a tree topology with one controller, three switches and four hosts. Controller is tagged with c0, switches are tagged with s1,s2 and s3 And hosts are tagged with h1,h2,h3 and h4. All the device are connected by links.

In our scenario POX controller comes with two network devices which are hub, layer2 learning switch. We used Xming X server for windows and PuTTY for multiple remote connections to virtual hosts. We are able to learn POX controller functionality, make modifications in code and perform experiment.

![Fig. 3: Mininet virtual network](image-url)
V. FLOW CHART

We created simple flowcharts for packet processing at switch level in Fig.4. When packet arrives from any host to switch, first switch match the incoming packet with rules installed in flow table. If rules are matched then packet is going to controller otherwise it can be drop from switch. So only legitimate packet will go to controller, traffic congestion is reduced. Layer2 learning switch use MAC address and port number. In Layer2 learning switch header information in received packet was checked against firewall rules. The packet is either rejected or allowed based on firewall rule.

When working Mininet initially we create remote SSH connection with virtual machine. Xming X server and PuTTY were used with X11 forwarding enabled. We use three remote connections: one for Mininet, other for adding flow entry to switch and last is use for POX controller.

VI. RESULT

Initially we learn layer2 learning switch from Coursera[6]. After we modified that code and add firewall entries in that python code. We use Xterm utility to access each host, controller and switch individually using this command in Mininet:

$ xterm c0 s1 s2 s3 h1 h2 h3 h4

At each terminal we recorded MAC address. We learn entries installed by POX controller in layer2 switches. We have done number of experiment in which our implemented firewall code for layer2 switch showed successful network traffic. For example we select a case in which host h1 and h4 were allowed to communicate with each other and other communications were blocked. As shown in Fig.6 when POX controller was invoked, firewall rules were installed in three switches. Logging level was set by log level --DEBUG. So with we can say that from our experiment h1 and h4 host communicate with each other successfully. As shown in Fig.6 firewall rules is installed successfully at three switches which allowed communication between host h1 and h4.

VII. CONCLUSION

We have successfully implemented a firewall by modifying layer2 switch code in POX controller. A flowchart was developed which demonstrate that malicious traffic is detected at switch level rather than control plane. Traffic is successfully controlled by firewall rules. The result shows that firewall can protect our internal network. In future, we implement firewall which protect virtual network also.
REFERENCES


[10] www.sdncentral.com