Software-Defined Networking

Lab 7

SDN Security and REST-Python

University of Colorado Boulder

Department of Computer Science

Professor Levi Perigo, Ph.D.

# Lab Summary

This lab is intended to be an overview of basic TLS and firewall security in SDN. Securing the SDN controller is critical to the security of the entire SDN. TLS is the standard way to secure the southbound communication from the networking devices and the SDN controller. This lab will provide the fundamentals of implementing TLS for OpenFlow communications.

A firewall is a very critical application for any network. It acts as the first/last line of defense against any unauthorized user trying to access or exploit the network. Such users can cause much harm to the networks by adding/changing flow entries to cause misconfigurations, execute DDoS attacks or just silently sniff critical information of the network. The purpose of this lab is to implement a script on Floodlight controller using a python code and understand how rules are implemented to execute certain actions on the packets which match them. The experience gained from completing this lab should be used as a foundation to understanding higher layer firewalls and how software can control the network. In the future, this basic firewall can be enhanced with additional functionality.

# Objective 1 – Attack SDN controller

In this objective you will modify the scripts written before to attack an SDN controller, and detect and stop the attack.

1. Initialize Floodlight on the controllers VM.
2. Initialize a linear topology in Mininet with four switches, remote Floodlight controller, and OpenFlow v1.3.
3. Clone the scripts from your GitHub account you pushed for Lab 5 on the Mininet VM. Paste a screenshot of the commands used. [**1 point**]

## Attack:

1. To attack the controller, you have to modify the cloned script and execute it on the Mininet VM.
2. The objective is to detect the controller IP and the OpenFlow port it uses and initiate an attack using Scapy (or any other Python based tool you prefer).
3. The attack should be a Denial-of-Service by sending multiple Packet\_In messages to the controller’s IP and port. Please use the same source port for all your attack packets.
4. Paste screenshots of your script detecting the controller’s IP and port, and of the attack. [**20 points**]

## Detect and stop:

1. To detect the attack, use the script from Lab 5 to count the number of Packet\_In messages received from a switch IP:port.
2. You must execute the script on the controllers VM and display a message when a threshold (say more than 100 Packet\_In messages from one switch IP:port) are detected. Paste screenshots of your script detecting an attack to the controller. [**20 points**]
3. To stop the attack, your script should add an iptables on the controllers VM to block packets to the controller’s IP and port from the source port of the attack packets. Paste screenshots of the iptables rule added and confirm that the attack has been stopped. [**20 points**]
4. What is another way to prevent such attacks? [**1 point**]

After completing the above objectives, create a new branch in your GitHub repo and then push the modified scripts back to GitHub making them master. Paste screenshots of the commands you used. [**5 points**]

# Objective 2 – SDN Security using SSL/TLS

Now that you have successfully attacked the controller and detected and stopped the attack, in this objective you will be creating SSL/TLS connections between Mininet and SDN controller. For simplicity you will only need to use Mininet VM to do this work. SSL/TLS is useful to secure SDN systems, but it cannot be enabled by a single command.

1. Run the following commands inside Mininet VM to generate all the keys required for this lab:

*cd /etc/openvswitch*

*sudo ovs-pki req+sign ctl controller*

*sudo ovs-pki req+sign sc switch*

*sudo ovs-vsctl set-ssl \*

*/etc/openvswitch/sc-privkey.pem \*

*/etc/openvswitch/sc-cert.pem \*

*/var/lib/openvswitch/pki/controllerca/cacert.pem*

1. Provide screenshot of results. [**5 points**]
2. Explain what is cert.pem, privkey,pem, req.pem. [**15 points**]
3. In a separate window of the Mininet VM, run the following command:

*sudo ovs-controller -v pssl:6633 \*

*-p /etc/openvswitch/ctl-privkey.pem \*

*-c /etc/openvswitch/ctl-cert.pem \*

*-C /var/lib/openvswitch/pki/switchca/cacert.pem*

Explain what this command does and what the three options do. [**10 points**]

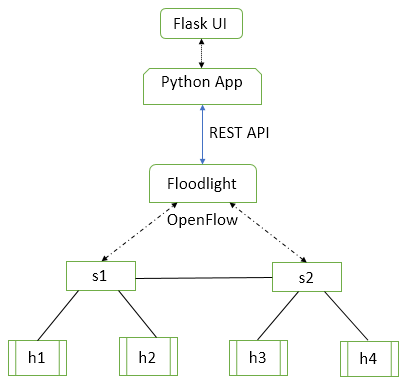
1. Create a basic topology with 1 controller, 1 switch and 3 hosts in MiniEdit. Export the .py file into the VM. Modify the .py file such that the OvS connects to the controller over a SSL connection. Paste screenshots of the topology and the modification to the .py file. [**15 points**]
2. Execute this .py file and paste screenshot that indicates the switch is connected to the controller via a SSL secure connection. [**5 points**]
3. Please describe the steps needed to create a SSL secure connection between a switch and a controller in your own words. [**5 points**]
4. Can you describe the types of attack this objective can help prevent? [**5 points**]

# Objective 3 – SSL/TLS on Floodlight

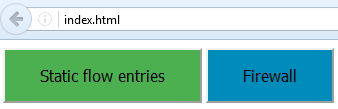
Complete the same objectives as in Obj 2 using Floodlight as the controller. Mention the steps you followed and paste screenshots indicating a successful SSL connection between OvS and the Floodlight controller. [**15 points**]

# Objective 4 – REST Static flow entries and Firewall via Python

In this objective, you will be writing a Python script that uses Flask (or any library of choice, but “Flask” will be used throughout this document) to create a GUI, takes inputs from the user via the GUI and uses the REST API on Floodlight to configure static flow entries and firewall rules.



1. Before initializing Floodlight, edit the file - /home/sdn/floodlight/src/main/resources/ floodlightdefault.properties file and remove the line - net.floodlightcontroller.forwarding.Forwarding,\. Can you explain what will removing this line do? [**5 points**]
2. Create the above topology in Mininet with two switches and two hosts connected to each switch and the remote Floodlight controller. Do a pingall. Will it work? Why/why not? Do you see flow entries on the switches? [**5 points**]
3. Write a Python script to create a simple REST client for accessing the controller’s REST API.
4. Use Flask to create a GUI, index.html should display 2 options – Static Routing and Firewall, and depending on what user selects redirect to another appropriate html page.



1. The static routing html page should take inputs from the user for these fields:

DPID, priority, In-Port, Eth-type, Dest IP, Action (flood or the particular port number)

You will have to add static flow entries on switches s1 and s2 to flood ARP packets, and forward other packets to the appropriate out port so that all hosts are able to ping each other.

Paste screenshots of the relevant flow entries on the switches. [**5 points**]

1. When the user selects Firewall, by default everything should be blocked.
2. The firewall html page should take inputs from the user for these fields:

DPID, priority, In-Port, Eth-type, Src IP, Dest IP, L4 protocol

You will have to add firewall rules to allow specific communication based on the user inputs. Do not use the Firewall API, add static flow entries for the traffic allowed.

1. Paste screenshots of the firewall rule added and the ping outputs indicating only the specific connections allowed.
2. To achieve full credit, attach the script along your submission and please show the functioning of your code to the TAs. [**80 points**]

## Total Points \_\_\_ / 237 points