Student Name: Tiriac Ioana-Raluca   
Email: itiriac@asu.edu  
Submission Date: 21.11.2021  
Class Name and Term: CSE548 Fall 2021

Project 3 : DoS attack on SDN controller

# Project Overview

This project requires students to simulate a Denial-of-Service attack on an SDN environment. The SDN environment is to be setup based on POX controller, OVS – Open Virtual Switch and containernet. To mitigate the DoS attacks, the provided L3Firewall.py file has been enhanced to detect and filter out basic IP Spoofing and MAC Spoofing attacks.

# Network Setup

A new VM has been created using the given vdi image and following lab CS-SYS-00101 again this VM has been setup with NAT network (10.0.2.0/24).

# Software

For the lab the following software has been used:

* Open vSwitch <https://www.openvswitch.org/>
* Mininet
* POX Controller
* ContainerNet
* packages installed: mininet, POX, OVS , Python & Python 3

# Project Description

Text

Description automatically generated *Check mininet installation*

Text

Description automatically generated *Check POX installation*

***Lab assessments***

1. ***(15 points) Setting up mininet and Running mininet topology  
   (a) Create a mininet based topology with 4 container hosts and one controller switches.*** 
   * ***Add link from controller to switch 1.***
   * ***Add link from switch 1 to container 1.***
   * ***Add link from switch 1 to container 2.***
   * ***Add link from switch 1 to container 3.***
   * ***Add link from switch 1 to container 4.***

***(b) Run the mininet topology. (Create mininet topology using mininet command-line)***

sudo ./pox.py openflow.of\_01 --port=6655 pox.forwarding.l3\_learning \

pox.forwarding.L3Firewall \

--l2config="l2firewall.config" \

--l3config="l3firewall.config" \

log.level --DEBUG

Next, we’ll create the required mininet environment consisting of 4 containernet hosts, one OVS switch and one controller running on port 6655 that will bind to the previously created POX controller.

mn –topo=single,4 –controller=remote, port=6655 –switch=ovsk –mac

Text

Description automatically generated

From the screenshots we can see the created links between the components.

1. ***(15 points) Should assign IP addresses to hosts.***

***Make the interfaces up and assign IP addresses to interfaces of container hosts: Assign IP address 192.168.2.10 to container host #1  
Assign IP address 192.168.2.20 to container host #2  
Assign IP address 192.168.2.30 to container host #3***

***Assign IP address 192.168.2.40 to container host #4***

Text

Description automatically generated

From the above screenshot we can see the container hosts created previously and their corresponding network interfaces.

Text

Description automatically generated

After using ifconfig to change each container host’s IP address, we can see the changes with xterm command:

Graphical user interface, text

Description automatically generated

1. ***(15 points) Perform Flood attack on SDN controller following a suggested procedure:***

###### **Run l3 learning application in POX controller.**

sudo ./pox.py openflow.of\_01 --port=6655 pox.forwarding.l3\_learning \

pox.forwarding.L3Firewall \

--l2config="l2firewall.config" \

--l3config="l3firewall.config" \

log.level –DEBUG

Text

Description automatically generated

*Running l3\_learning application in POX controller with Debug on*

***(b) Check openflow flow-entries on switch 1.***

***(c) Start flooding from any container host to container host #2. using source address 10.10.10.1 (d) Check Openflow flow entries at switch 1***

(d***) Check Openflow flow entries at switch 1***

In the next screenshots the 4 xterm CLIs of the container hosts will be aligned to demonstrate DoS attack by flooding the OVS. One host will show the OVS flows, one will flood another, and another will ping another container.

Text, timeline

Description automatically generated

In the screenshot above ovs-ofctl dump-flows s1 is not showing much except for the l2firewall.config rules, which is correct because there is no traffic yet.

Then from host1 we start flooding host2 and observing the ovs dump flows on s1. From host 2 we start pinging host4 and from host4 we also try pinging host2 – the flooded host. The high packet loss is easy to observe in the communication between host2 and host 4, caused by the flooding of h2.

After the flooding the OVS slowly comes back to normal, it needs a few minutes to resume normal traffic.

Text

Description automatically generated *OVS flow dumps \*during\* flooding*

Text

Description automatically generated *OVS flow dumps near* *end of flooding and during PING-ing between h2 and h4*

1. ***(55 points) Mitigate DoS attack by implementing port security and using OpenFlow based firewall.*** 
   * ***(25 points) You should illustrate (through screenshots and descriptions) your implemented pro- gram codes.***
   * ***(15 points) You should demo how your implementation can mitigate the DoS through a sequence of screenshots with explanation.***
   * ***(15 points) You should submit the source codes of your implementation.***

The last part of the lab was implemented by working with L3Firewall.py file and adding code to it. For the purpose of storing DoS attacks information and MAC and IP spoofing attempts two dictionaries objects were used in code – Spoofing Dictionary and Blocked Dictionary.

**The Spoofing dictionary** object memorizes all OVS flows, by using the MAC address as a key and the source & destination IP address as values, but also the OVS port. **The Blocked Dictionary** is used as intermediary data structure to transfer all blocked flows into the l3firewall.config file, thus blocking all prohibited flows.

Most of the logic happens inside enforcePortSec() function, which is called for each OVS flow, through the \_handle\_PacketIn(self, event) function of the L3 firewall file.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with medium confidence

*Handling both MAC and IP spoofing inside enforcePortSec()*

Any spoofing attempt is blocked by calling addRuleToL3Config, which appends the MAC, src ip, src dest & so on details to the l3firewall.config to block the flow in future attempts/packages to spoof the OVS.

***IP Spoofing Mitigation***

*IP Spoofing* happens when the attacker uses the same MAC address to spoof a target IP address, but uses spoofing/different source IPs to attack his target. *MAC Spoofing* on the other side happens when the attacker has a constant source IP address, but spoofs random different MAC addresses to target his host.

We’ll start flooding h2 container with IP 192.168.2.20 from h1 and at the same time ping it from another host h4. The packet transmission is not affected by the flooding:

Text

Description automatically generated

Because the POX controller is running in DEBUG it’s picking up L3Firewall’s debug messages and detecting IP spoofing :

Text

Description automatically generated

# Appendix B: Attached files

* L3Firewall.py

# References

* Open vSwitch Cheat Sheet: <https://therandomsecurityguy.com/openvswitch-cheat-sheet/>
* Containernet: <https://containernet.github.io/>
* Containernet tutorial: <https://github.com/containernet/containernet/wiki/Tutorial:-Getting-Started>
* Port security: <https://packetlife.net/blog/2010/may/3/port-security/>