**CARNEGIE MELLON UNIVERSITY - AFRICA**



**NETWORK SECURITY PROJECT 1**

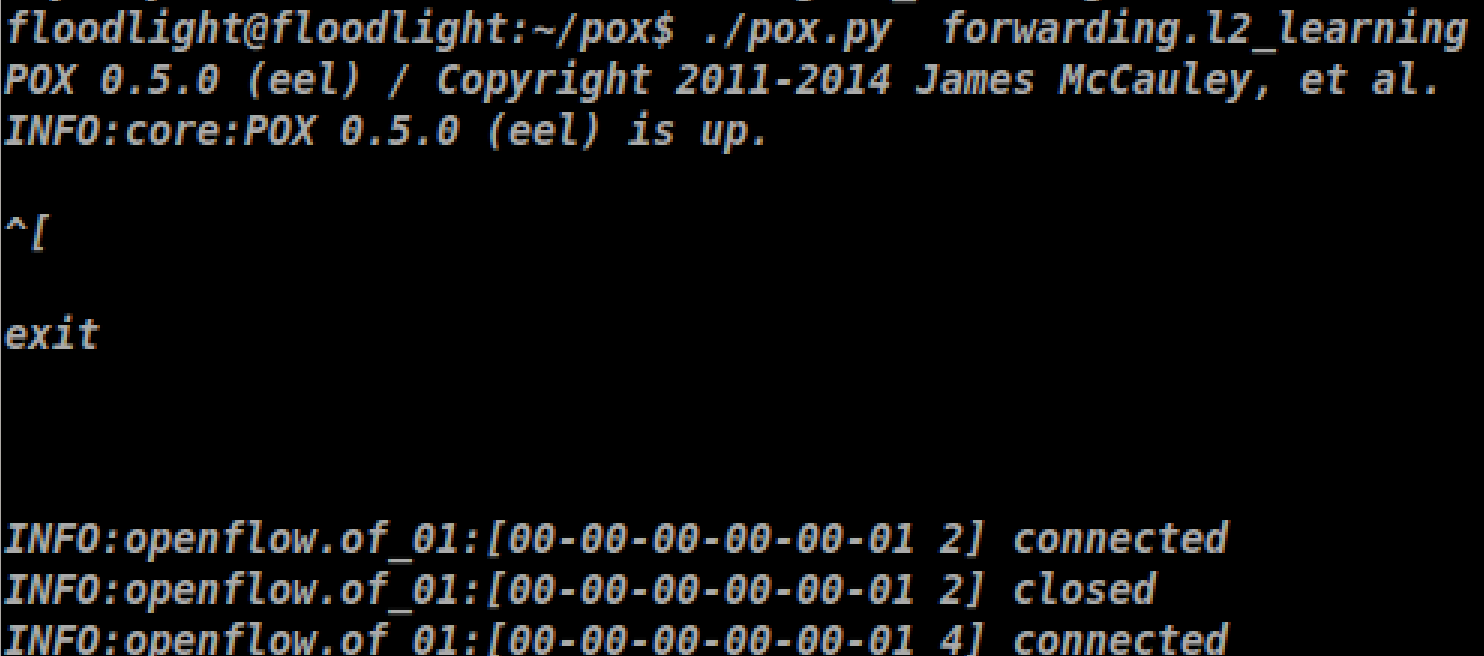
Abigail Wooley

Samukeliso Mabarani

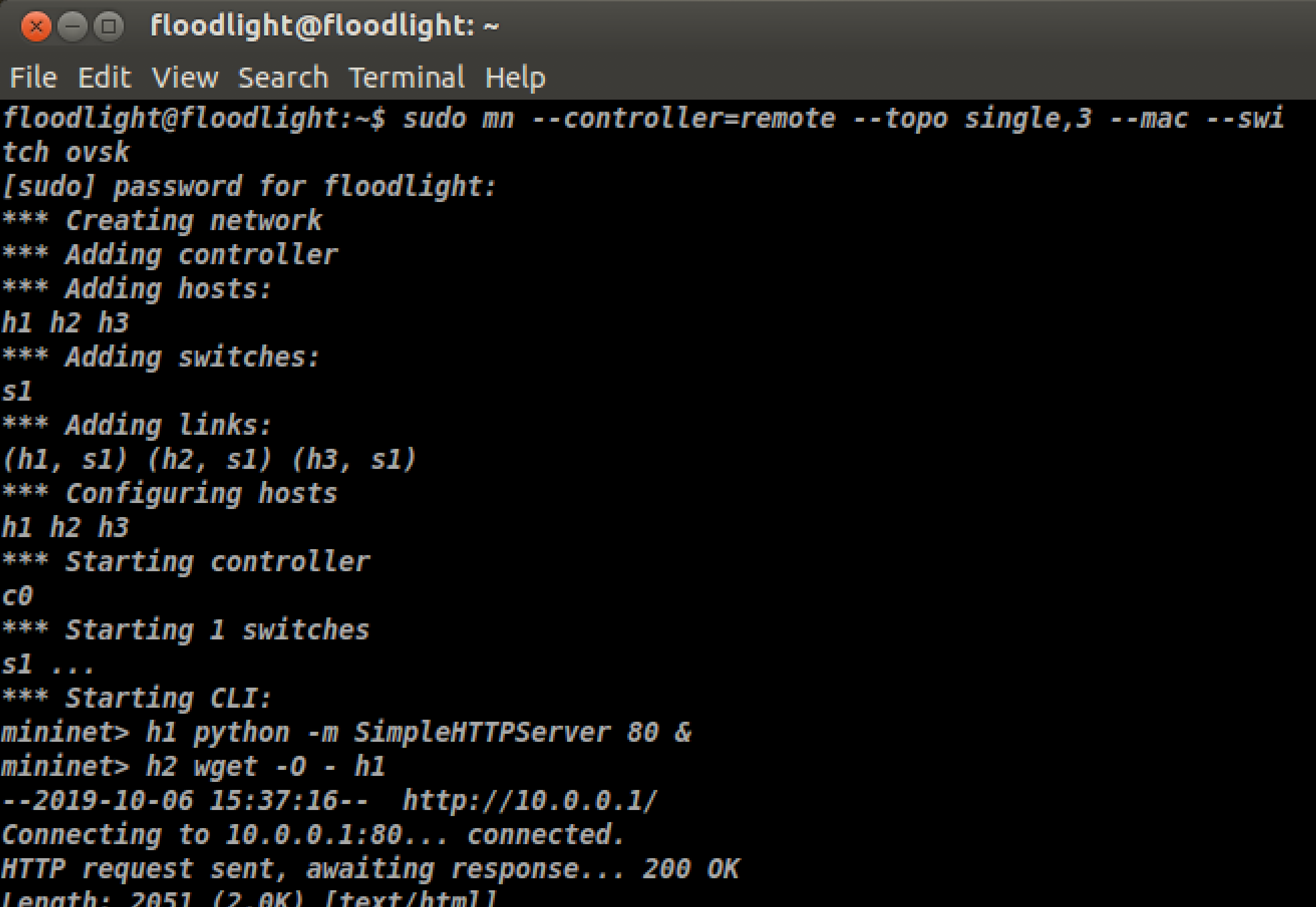
1. Use the L2 learning switch in POX at `pox/forwarding/l2 learning.py' (Reactive Mode); refer to the useful

links section for more information.

Having installed all the software required to start this project, the next thing we did was to launch mining using the command ***sudo mn***, then we started the POX controller by using the command ***sudo mn --controller = remote --topo single --mac --switch ovsk***. The result of this can be seen the screenshot below:



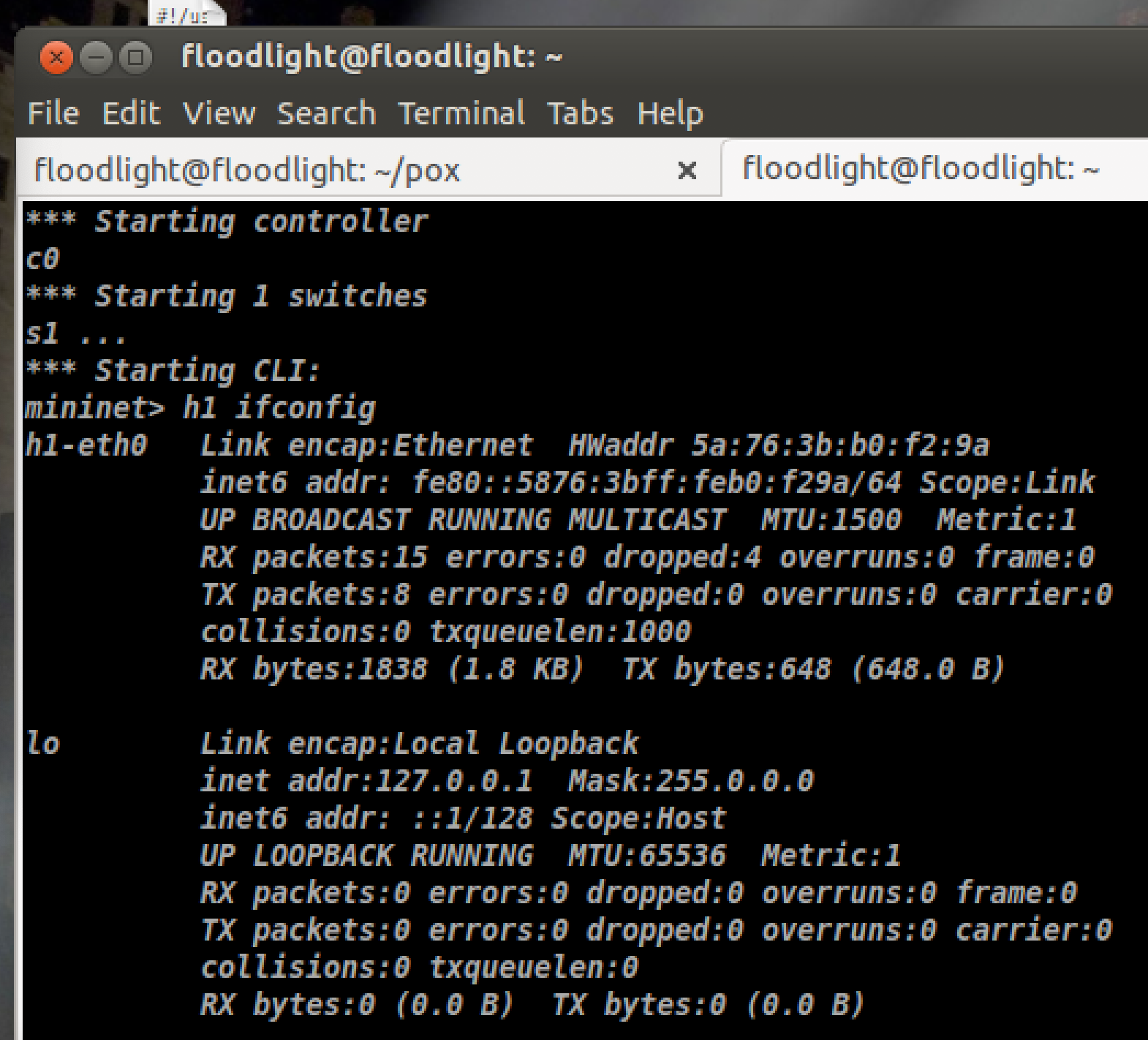
The command **Sudo mn** launches mininet, **--controller=remote** specifies the started POX controller as the controller to be used (remote controller. This is used in play of specifying the IP address and port number for the device to be used). The command **--switch ovsk** specifies that the open switch should be used. **--topo single,3 --mac** specifies that 3 hosts are required and hence this is created and each linked to the switch as can be seen in the terminal below:



2. Start a simple HTTP server on h1 (Host 1). HINT: Refer to the useful links; a one line command would

solve this problem.

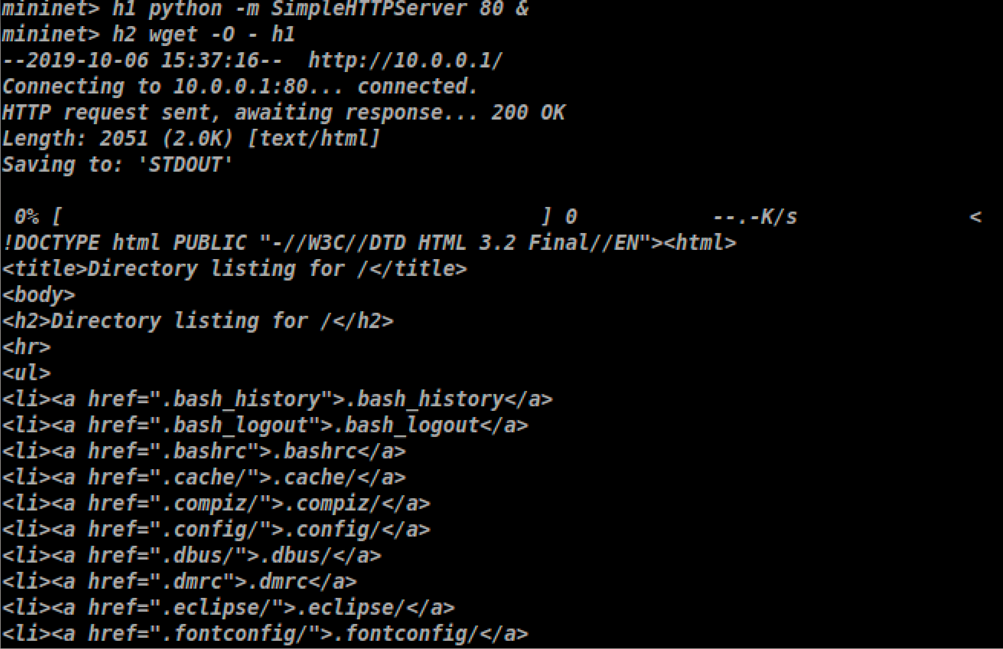
The next step of this configuration is to start a simple HTTP server on h1 and then execute a HTTP GET request from h2 to h1 as shown below:



3. Without executing any attack, execute a HTTP GET request from h2 to h1. The request should complete

successfully.

**h2 wget -O - h1** is a command which executes a GET request from h2 to h1and displays the HTML code to standard output as can be seen in the screenshot below:.



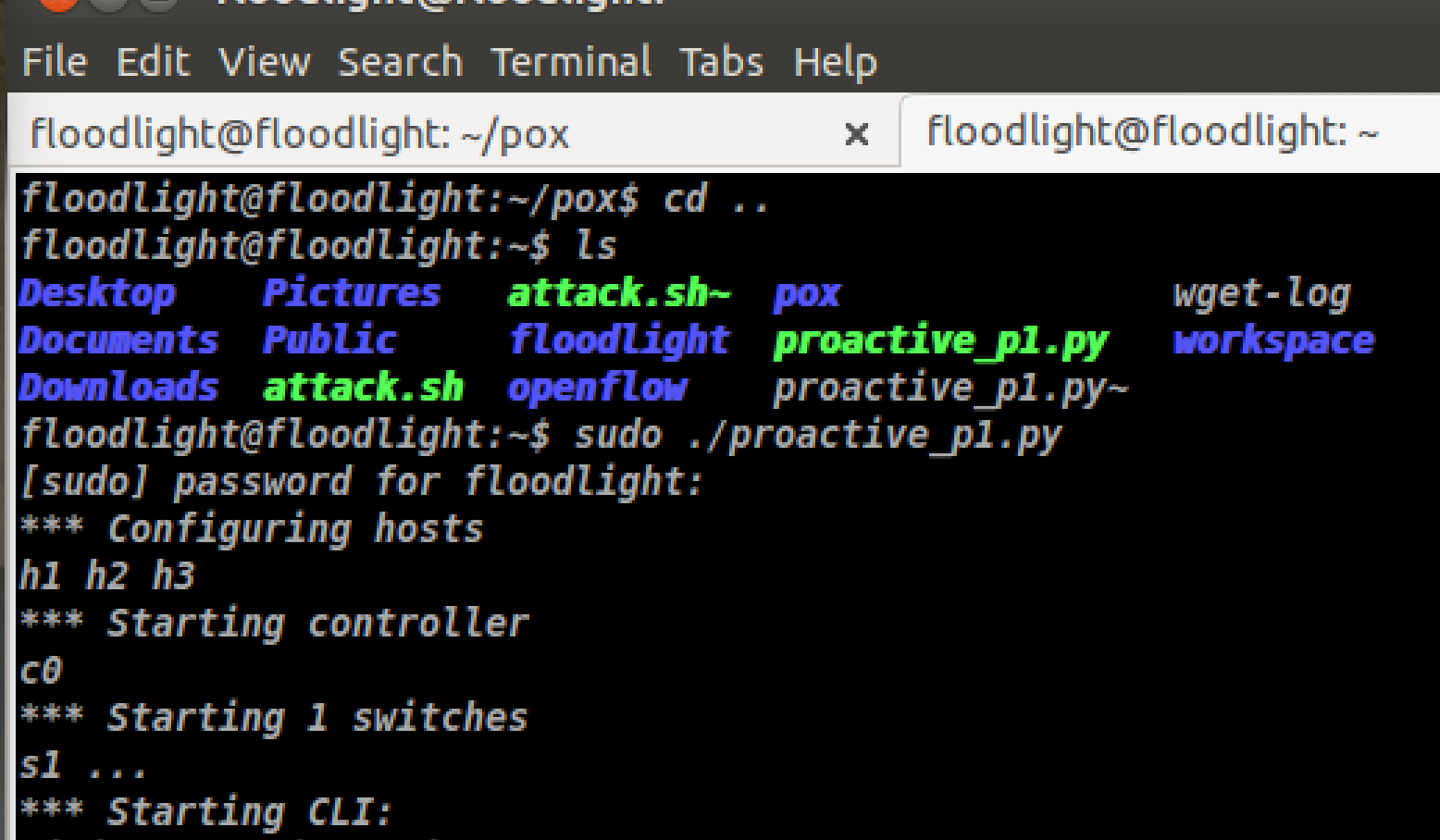
4. Now, launch a SYN flood attack from h3. You can use known tools or write a SYN flood attack by yourself.

For a successful attack, note that you should configure your SYN flood attack to randomized addresses to

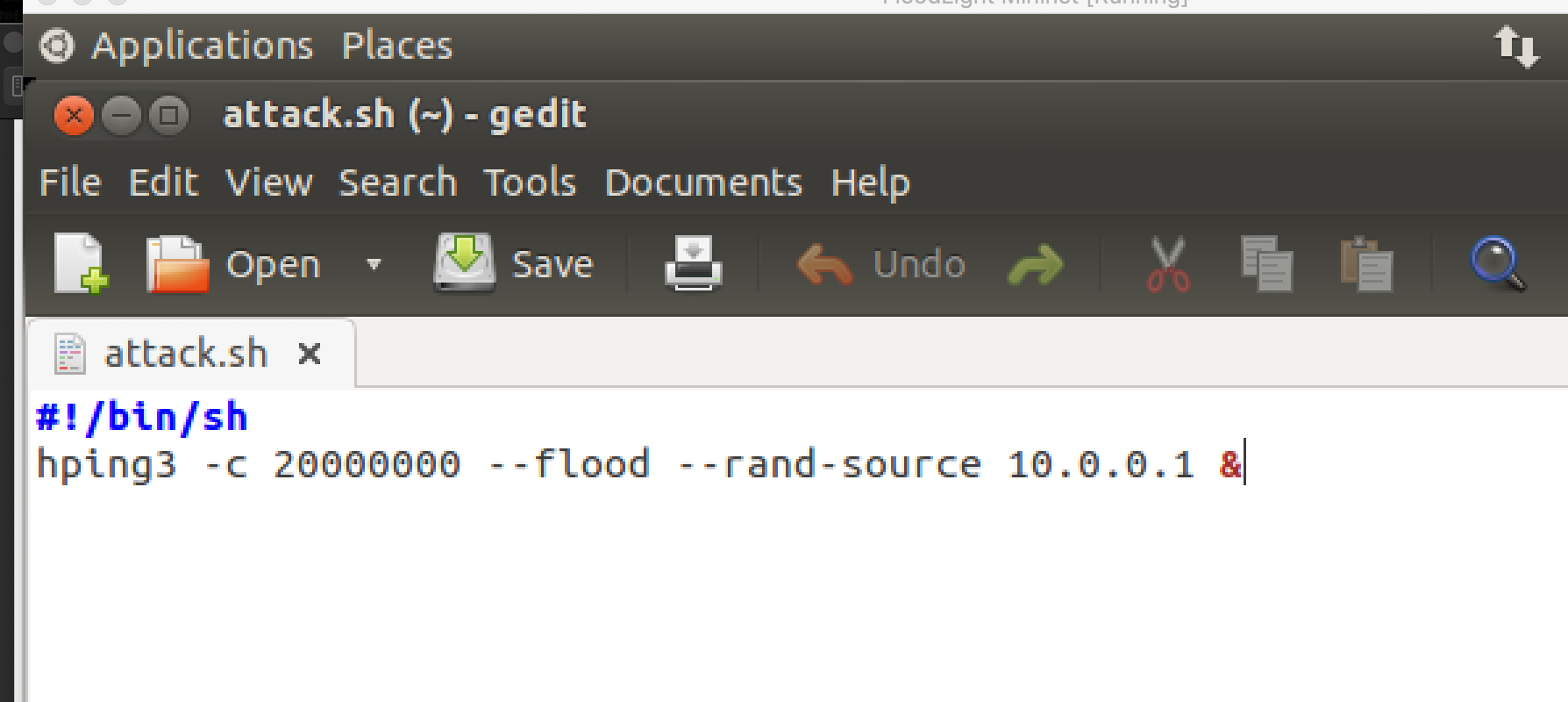
ensure that the switch cannot cache" these ow rules so that it always talks to the controller and receives a

new flow table entry. (Write your attack script in a file named `attack.sh')

The screenshot below shows the attack.sh file that we created.

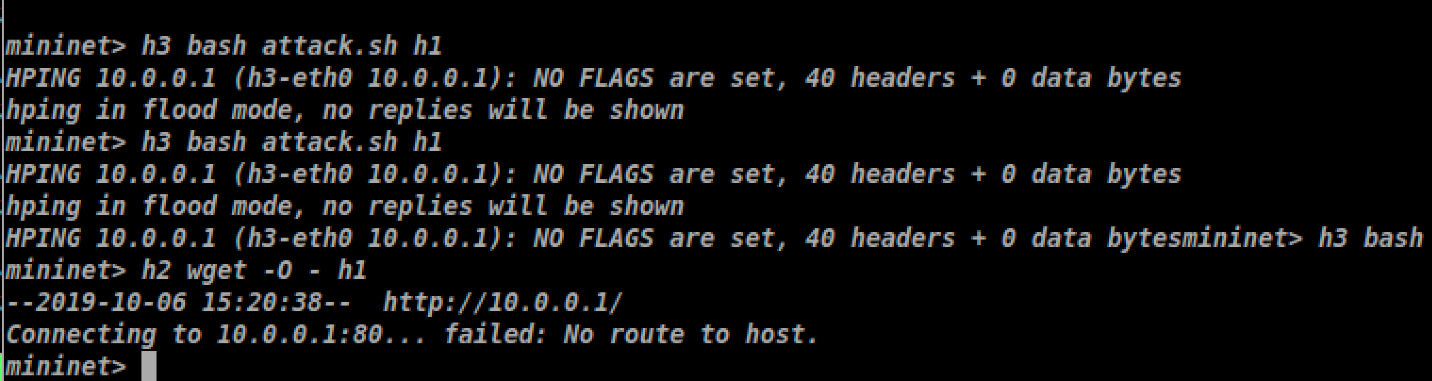


To launch the SYN flood attack form h3, we wrote a bash script and saved in a file called attack.sh. as shown below:

# **hping3 -c 20000000 -d --flood --rand-source** **10.0.0.1 &**

We’re sending **20000000 packets** (**-c 200000000**). To direct the attack to our victum’s HTTP web server we specify **port 80** (**-p 80**) which is the default port for HTTP and use the **--flood** flag to send packets as fast as possible. The **--rand-source** flag generates spoofed IP addresses to disguise the real source and avoid detection but at the same time stop the victim’s **SYN-ACK reply packets** from reaching the attacker.

5. After launching the attack, try to execute a HTTP GET request from h2 to h1 again. Report your result.

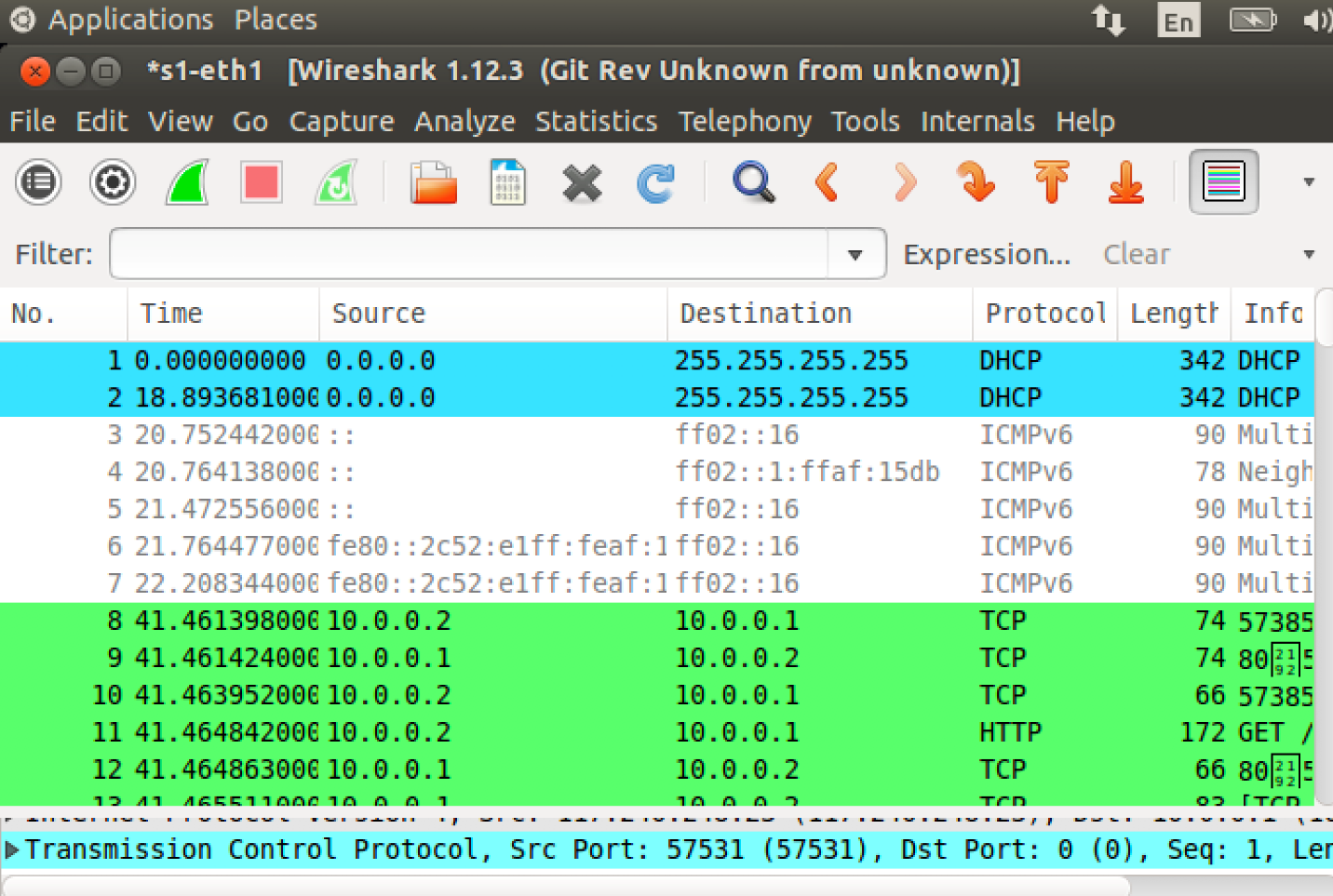


When we tried to execute an HTTP GET request from h2 to h1 again we failed to reach h1. The connection to h1 failed. This is because h1 is flooded and any other connection attempt will fail to successfully connect. The “no route to host” indicates that means that h2 can't reach the target, h1.

6. Dump the flows on the switch. You may also want to observe the number of requests to the controller using

tcpdump. (HINT: You may capture traffic on the relevant interface for a specific amount of time. This can

help you compare your findings with those obtained using proactive controller)



We used Wireshark to detect the attack. As depicted, there’s a high volume of SYN packets with very little variance in time. Each SYN packet is from a different source IP address with a destination port 80 (HTTP), identical length of 120 and window size (64). If we filter with **tcp.flags.syn == 1** and **tcp.flags.ack == 1** we can see that the number of SYN/ACKs is comparatively very small. A definite sign of a TCP SYN attack.