LAB 6: FIREWALL

Objective:

To implement a Firewall in a star topology and block the TCP traffic based on IP address.

This lab is to demonstrate the Software Defined Network (SDN) concept for Layer 4 Firewall function using Open Flow protocol. Securing the SDN controller is critical to the security of the entire SDN.

Firewall: 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 several harms to the networks by adding/changing flow entries to cause misconfigurations, execute DDoS attacks or silently sniff critical information of the network. The purpose of this project is to implement a simple Layer 4 firewall on Ryu controller using python code and understand how rules are applied to execute specific actions on the packets which match them. The experience gained from understanding this lab should be used as a foundation for understanding higher layer firewalls and how software can control the network. In the future, this basic firewall can be enhanced with additional functionality.

Task: To demonstrate the Layer 4 firewall using Ryu controller. This firewall should allow TCP traffic based on 4 tuples (src_ip, dst_ip, src_port, dst_port) that you provide as an input via a csv or a config file pre-defined within the firewall application. Allowing only TCP traffic is a condition you should not overlook. To modify the firewall rules, the config file must be edited, and the RYU firewall app must be restarted. The firewall app code does not have to be edited for a change in firewall rules. Only TCP should be blocked, UDP and ICMP should be able to pass through.

PART 1:

Step 1: We are going to use two files for this lab viz OneFirewall.py and Allowance.csv. The file named 'OneFirewall.py' is the one we going to use to run the controller. Make sure you move that file in the **</usr/local/lib/python2.7/dist-packages/ryu/app/>** where the other RYU-controllers programs are stored like SimpleSwitch.py. Also, move the configuration file named 'Allowance.csv' in your home directory.

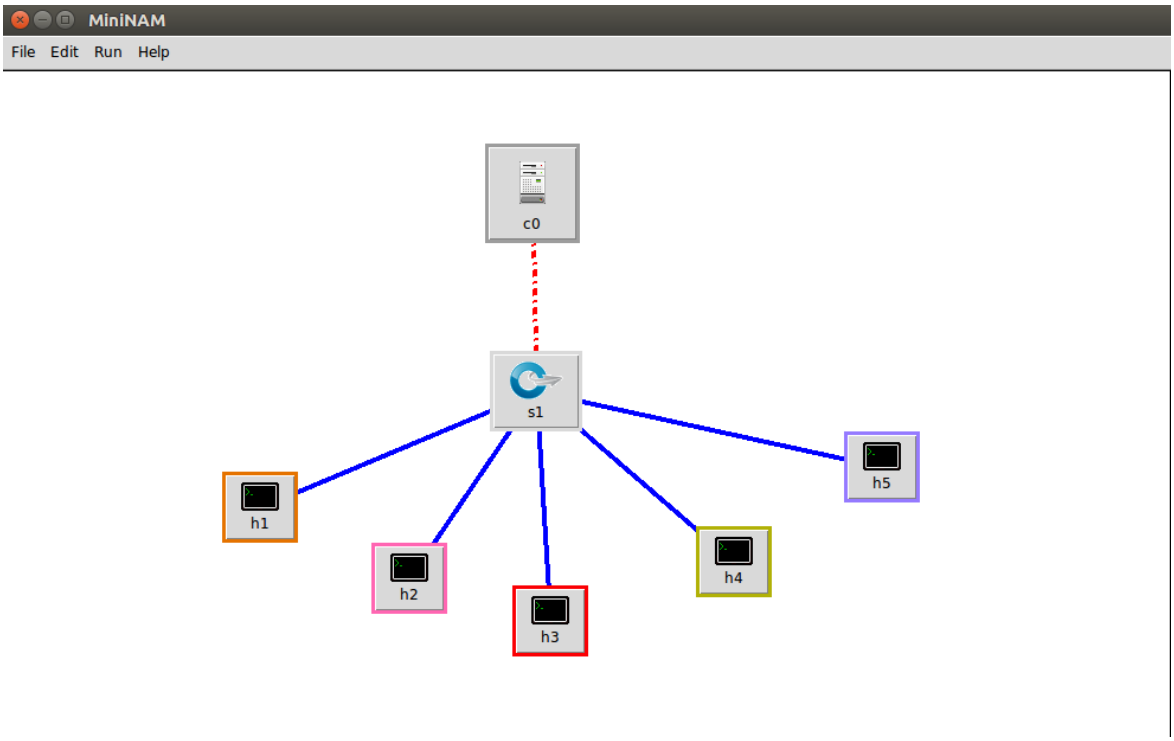
```
martand@martand-VirtualBox:~$ pwd
/home/martand
martand@martand-VirtualBox:~$ ls
Allowance.csv     lab1.mn  mininet          Pictures      test.py
Desktop           lab1.py  Music            Public        Untitled Document
Documents         lab2.mn  Performance1.py  ryu           Videos
Downloads         lab2.py  Performance.py   SDN-Firewall
examples.desktop  MiniNAM  perfor.py        Templates
martand@martand-VirtualBox:~$
```

Step 2: Now start the Ryu-controller along with the OneFirewall.py program.

ryu-manager ryu.app.OneFirewall --observe-links --verbose

Step 3: Now create a simple Mininet topology using one switch and five hosts.

sudo python MiniNAM.py --controller remote --topo single,5 --ipbase 10.0.0.1/24 --mac



(**Paste the screen-shot of the Topology**)

Step 4: Execute a pingall from Mininet CLI and **paste the screen-shot.**

The configuration file (Allowance.csv) is configured to allow TCP traffic between HTTP port of host 1 (having IP Address 10.0.0.1) & host 2 (IP Address 10.0.0.2) and between HTTP port of host 1 & host 3 (IP Address 10.0.0.3).

Step 5: Now start the HTTP server on the host 1 using Mininet CLI

h1 python –m SimpleHTTPServer 80

Next execute <**h2 wget h1**> and <**h3 wget h1**>. You will see that h2 & h3 client receives HTTP traffic. **(Paste the screenshot of both the HTTP traffic commands)**

Step 6: Next try executing <**h4 wget h1**>. (**Paste the screenshot of it and explain why it failed**)

Now close the topology and kill all the controllers and clear cache of Mininet.

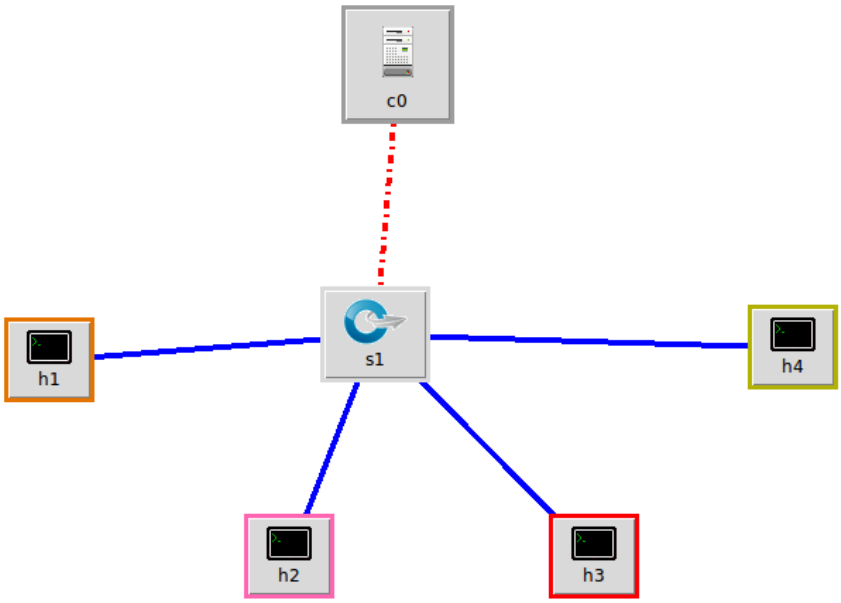
PART 2:

The next method we are going to try are using manually adding firewall rules.

Step 1: Start the Ryu-controller along with the rest_firewall.py program.

ryu-manager ryu.app.rest_firewall --observe-links –verbose

Step 2: Create a topology with one switch and four switches and make sure you add remote controller as well. (**Paste the screenshot of it.**)



Step 3: Enable the firewall on the switch.

s1 curl -X PUT http://localhost:8080/firewall/module/enable/0000000000000001

```
mininet> s1 curl -X PUT http://localhost:8080/firewall/module/enable/0000000000000001
[{"switch_id": "0000000000000001", "command_result": {"result": "success", "details": "firewall running."}
}]mininet>
```

Now we check the status of the firewall.

s1 curl http://localhost:8080/firewall/module/status

```
mininet>
mininet> s1 curl http://localhost:8080/firewall/module/status
[{"status": "enable", "switch_id": "0000000000000001"}]mininet>
mininet>
```

(**Paste the screenshot of it**)

Step 4: Add a rule to permit pinging between h1 and h2. You need to add the rule for both ways. Let's add the following rules. Rule ID is assigned automatically. We need to add rules between H1 & H2 and H3 & H4. Make sure there shouldn't be connectivity between H1 & H3 and H2 & H4.

curl -X POST -d '{"nw_src": "10.0.0.1/32", "nw_dst": "10.0.0.2/32", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001

```
mininet>
mininet> s1 curl -X POST -d '{"nw_src": "10.0.0.1/32", "nw_dst": "10.0.0.2/32", "nw_proto": "ICMP"}' http://
localhost:8080/firewall/rules/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_i
d=1"}]}]mininet>
mininet>
```

curl -X POST -d '{"nw_src": "10.0.0.2/32", "nw_dst": "10.0.0.1/32", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001

```
mininet>
mininet> s1 curl -X POST -d '{"nw_src": "10.0.0.2/32", "nw_dst": "10.0.0.1/32", "nw_proto": "ICMP"}' http://
localhost:8080/firewall/rules/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_i
d=2"}]}]mininet>
mininet>
```

Similarly, add rules between H3 and H4 and **paste the screenshot of them.**

Step 5: Now ping from H1 to H2 and H2 to H3 and H3 to H4 and **paste the screenshot of them.**

This concludes Lab 6.

Thank you.