Project 2 - Floodlight Firewall App

Firstly, to setup SDN environment we install Floodlight virtual box application from official website of Floodlight. Then after setting up floodlight virtual machine inside virtual box, we start to setup floodlight environment, enable firewall application (ACL REST API), and test Firewall REST API on our system through sequence of commands explained below.

1. Firstly, we use `ant` command to build all the java files inside the floodlight directory, Then, we use `java -jar target/floodlight.jar` to run the Floodlight within the VM.

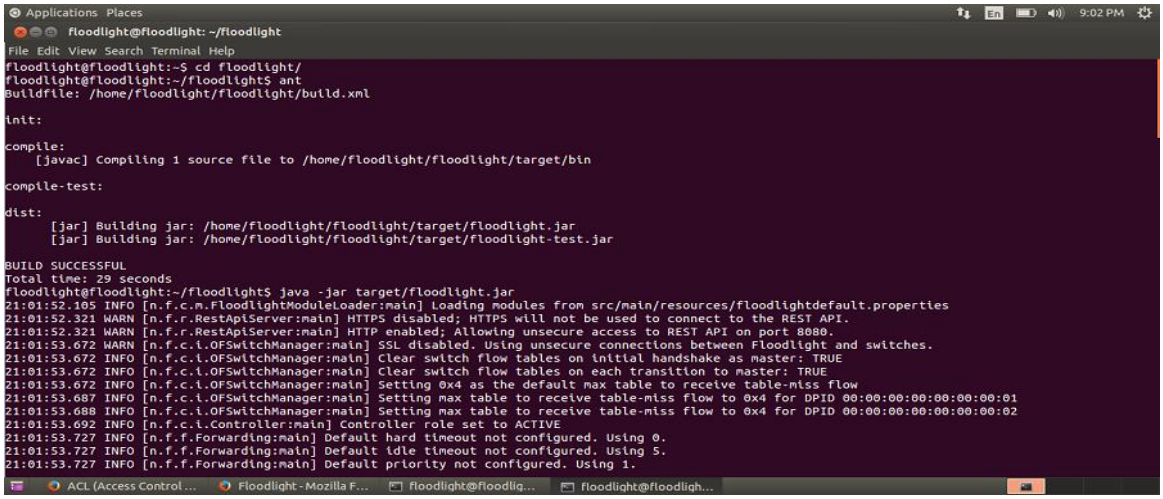

Fig. 1: Build Java files inside floodlight directory

a) Then, using command `java -jar target/floodlight.jar` command shown above, we get our localhost setup complete as shown below in ss.

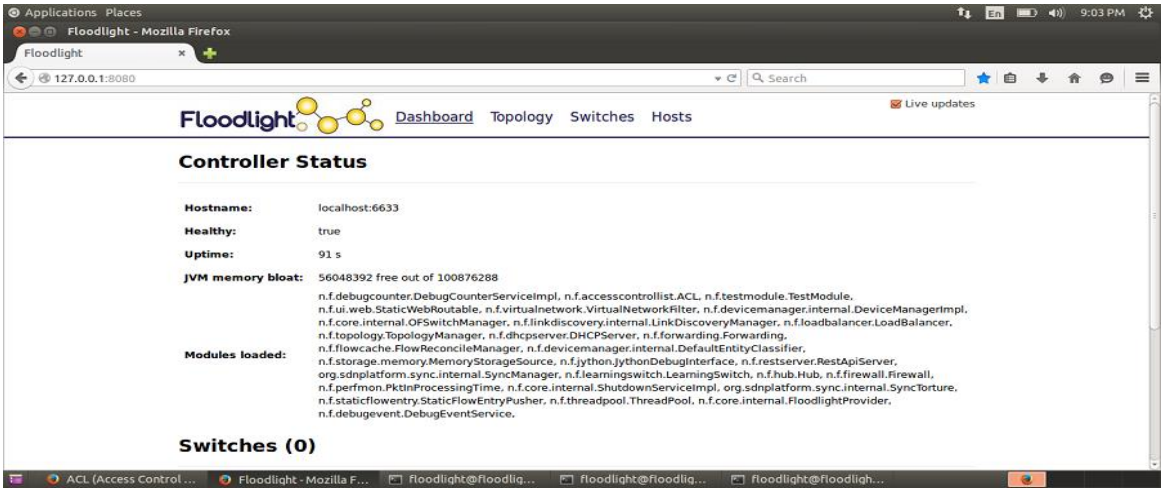

Fig. 1a: Checking localhost

Shivam Pandit

2. Now, since floodlight is running we will attach it to OpenFlow network. We used `Mininet` which is one of the best tools for this purpose. Then, we used mininet against remote controller using command:

```
sudo mn - -arp - -controller=remote,ip=10.0.2.15,port=6653 -switch
ovsk.protocol=OpenFlow13 -topo single,6
```

A single topology consists of a single switch connected to a number of hosts as specified in the topology build command. Here, we are using a single topology with 6 hosts connected to the switch.

Fig.2: Using Mininnet tool to implement OpenFlow

a) Then using `pingall` command we check if hosts are communicating with each other through switch as shown below.

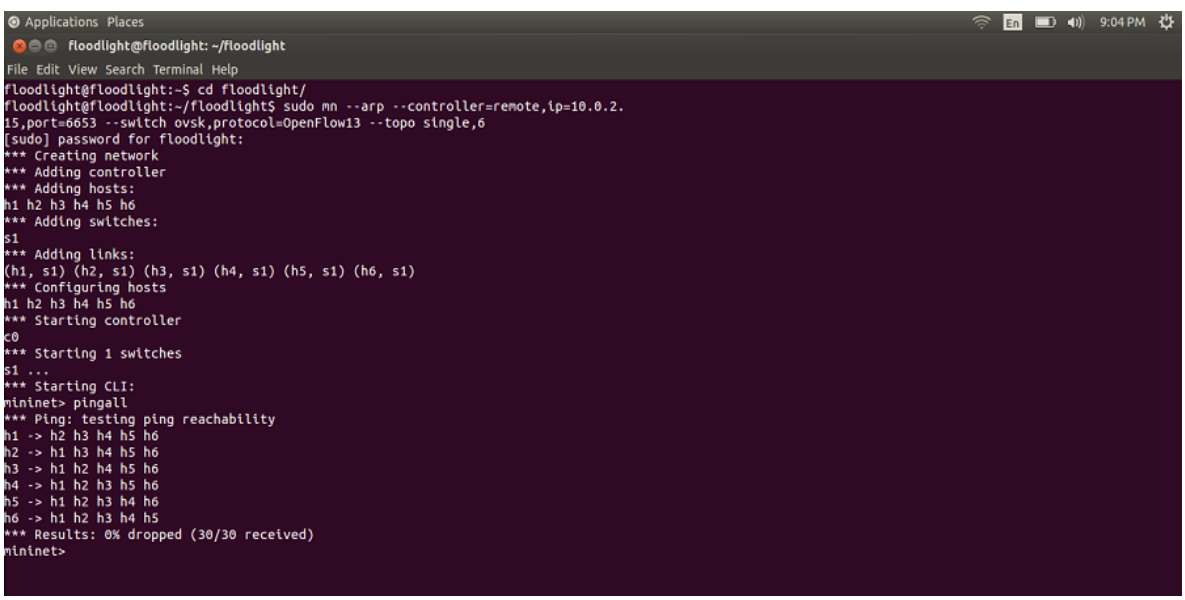

Fig 2.a: Using pingall command

Now we, can even see the hosts, switches as well as topology of the network in localhost as shown below.

b) Below SS shows the single topology with 6 hosts and one switch.

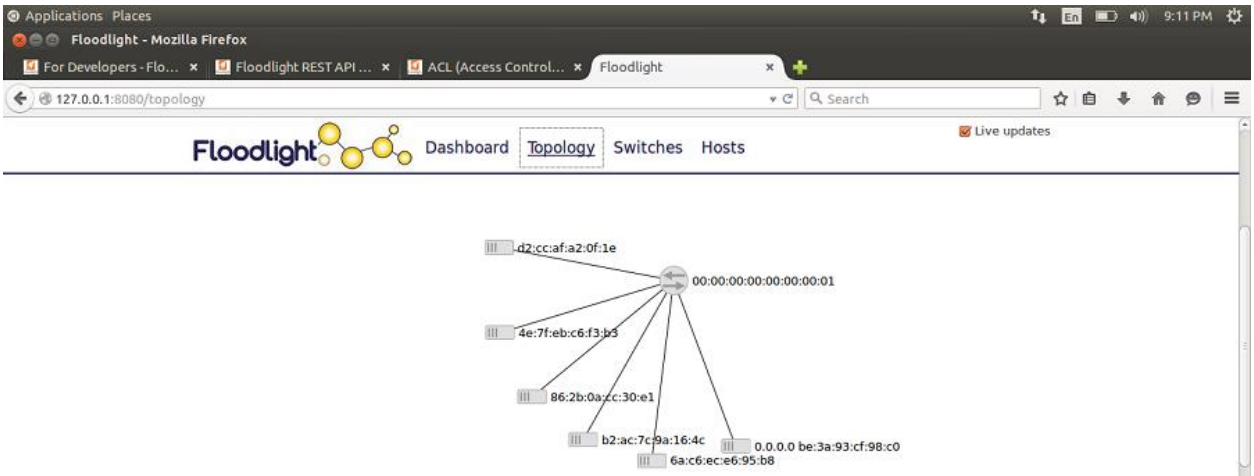

Fig. 2.b: Checking Topology in localhost

c) Below SS shows the DPID, ipaddress and other details of the switches and the hosts.

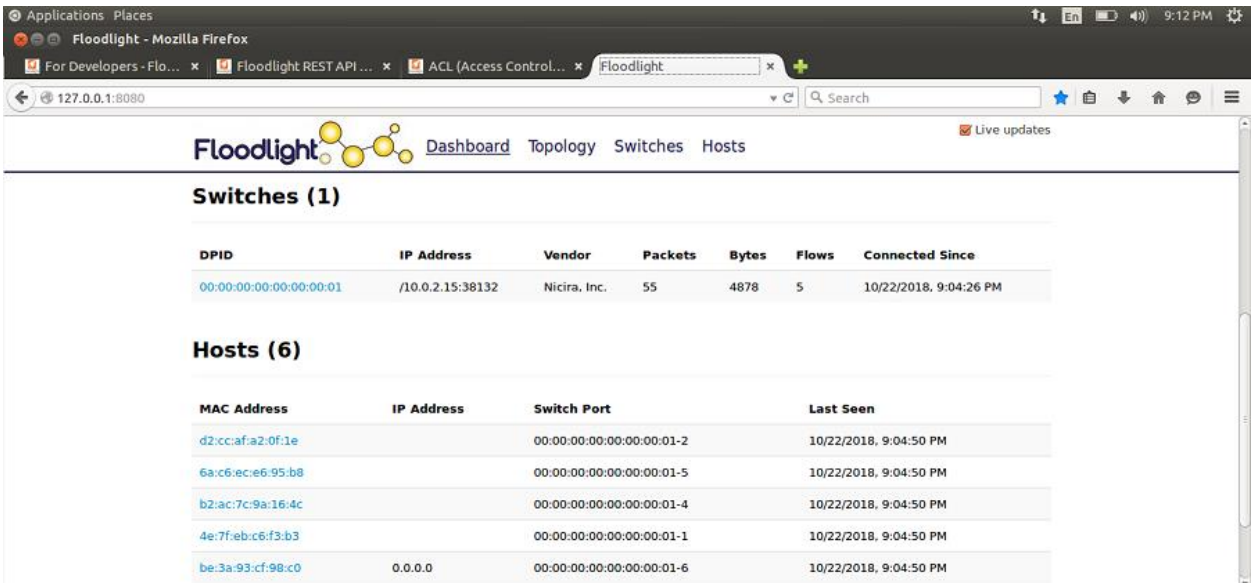

Fig.2.c: Checking Hosts and Switches in localhost

Now we, will be executing FIREWALL REST API and Floodlight REST API calls

Using ACL REST API's

1. **Using CURL to access REST API:** An example REST call that retrieves data from Floodlight at IP address <controller-ip> i.e. 10.0.2.15 is:

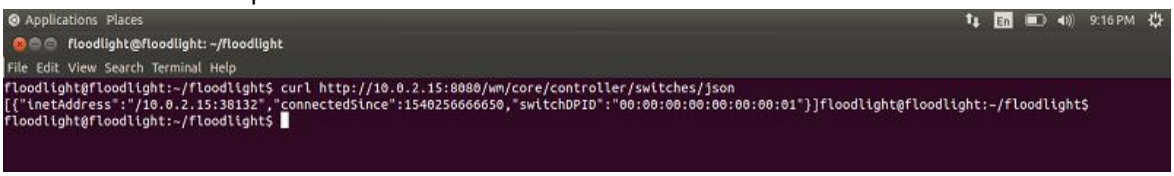

Fig.1: Using CURL to access REST APIs

2. **Adding an ACL Rule:** Now, we will add an ACL rule through ACL REST Interface using command

```
curl -X POST -d '{"src-ip":"10.0.0.1/32","dst-ip":"10.0.0.2/32","action":"deny"}' http://10.0.2.15:8080/wm/acl/rules/json
```

This command states that any traffic from source ip:10.0.0.1 to destination ip:10.0.0.2 will be denied by the switch.

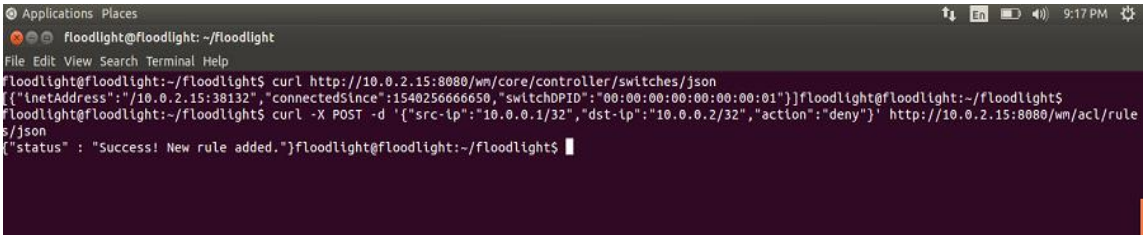

Fig.2: Adding an ACL Rule

3. **Listing All ACL Rules:** To list all the ACL rules currently active on the switch, we type the command

```
curl http://10.0.2.15:8080/wm/acl/rules/json | python -mjson.tool
```

This lists all the ACL rules currently active as shown below.

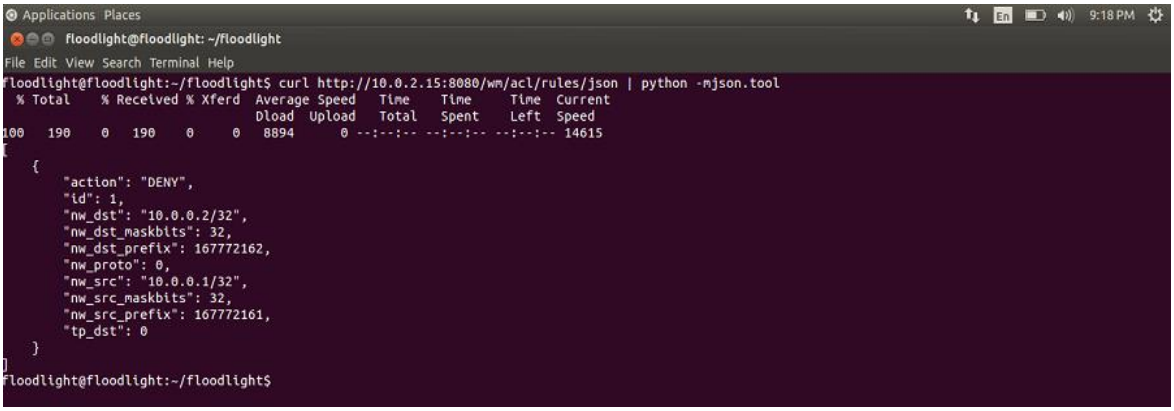

Fig.3: Listing all ACL Rules

4. **Removing ACL Rule:** To remove ACL rule, we use command `curl -X DELETE -d '{"ruleid":"1"}' http://10.0.2.15:8080/wm/acl/rules/json`

After executing the command, it deletes the ACL rule with a prompt Success! Rule deleted

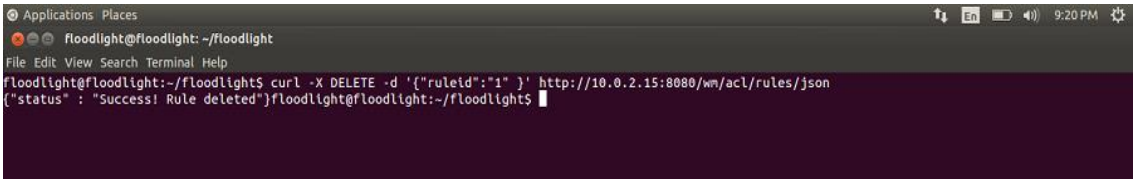

Fig.4: Removing ACL Rule

Examples Using CURL – Firewall REST API

1. **Check Status of Firewall:** To check the status of Firewall if it is enabled or disabled, we used command `curl http://localhost:8080/wm/firewall/module/status/json`

After executing the command, it shows that firewall is disabled in the terminal.

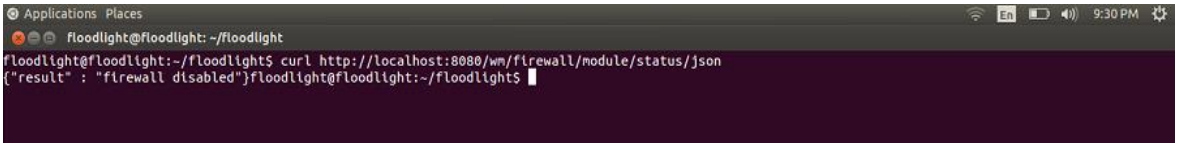

Fig.1: Check Status of Firewall

2. **Enable the Firewall:** Now, to enable the firewall using command

`curl http://localhost:8080/wm/firewall/module/enable/json -X PUT -d ''`

After executing the command, we rechecked the status of firewall and it showed enabled as shown below.

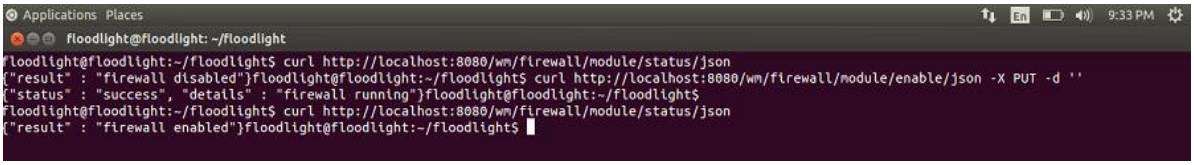

Fig.2: Enable the Firewall

3. **<u>Adding ALLOW Rule for flows through Switch: :</u>** To add ALLOW rule for all flows through switch we used the command

```
curl -X POST -d'{"switchid": "00:00:00:00:00:00:00:01"}'
http://localhost:8080/wm/firewall/rules/json
```

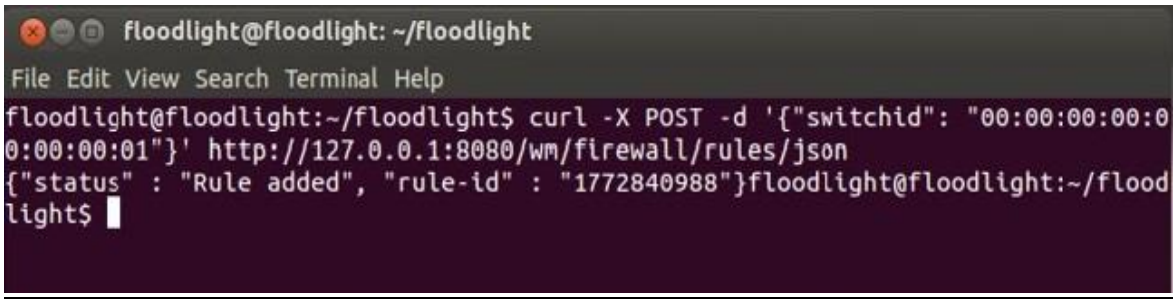

Fig.3: Adding ALLOW rule for flow through Switch

4. **<u>Adding ALLOW Rule for flows between two ip addresses:</u>** To add ALLOW rule for all flows between IP hosts. Not specifying action implies ALLOW rule. We used command

```
curl -X POST -d '{"src-ip": "10.0.0.3/32", "dst-ip": "10.0.0.7/32"}'
http://localhost:8080/wm/firewall/rules/json
curl -X POST -d '{"src-ip": "10.0.0.7/32", "dst-ip": "10.0.0.3/32"}'
http://localhost:8080/wm/firewall/rules/json
```

After executing the command, it shows prompt: Rule added.

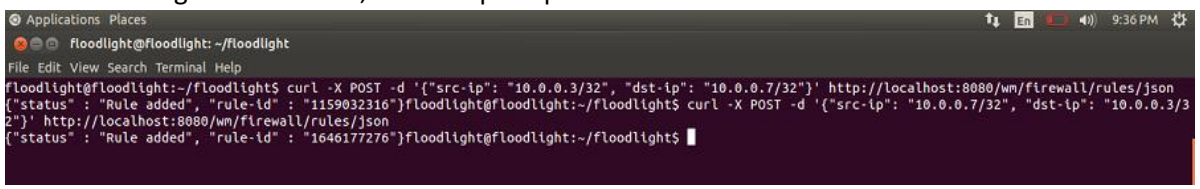

Fig.4: Adding ALLOW Rule for flow between two IP addresses

5. **<u>Adding ALLOW rule between two MAC addresses:</u>** To add ALLOW rule for all flows between host mac 00:00:00:00:00:0a and host 00:00:00:00:00:0b we use the command

```
curl -X POST -d '{"src-mac": "00:00:00:00:00:0a", "dst-mac":
"00:00:00:00:00:0a"}' http://localhost:8080/wm/firewall/rules/json
curl -X POST -d '{"src-mac": "00:00:00:00:00:0b", "dst-mac":
"00:00:00:00:00:0b"}' http://localhost:8080/wm/firewall/rules/json
```

After executing the command, it shows prompt: Rule added.

Fig.5: Adding ALLOW Rule between two MAC addresses

6. **<u>Adding an ALLOW rule for ping to work between IP hosts</u>:** To add ALLOW rule for ping to work between ip hosts we used the command

```
curl -X POST -d '{"src-ip": "10.0.0.3/32", "dst-ip": "10.0.0.7/32", "dl-type":"ARP"}' http://localhost:8080/wm/firewall/rules/json
curl -X POST -d '{"src-ip": "10.0.0.7/32", "dst-ip": "10.0.0.3/32", "dl-type":"ARP"}' http://localhost:8080/wm/firewall/rules/json
curl -X POST -d '{"src-ip": "10.0.0.3/32", "dst-ip": "10.0.0.7/32", "nw-proto":"ICMP"}' http://localhost:8080/wm/firewall/rules/json
curl -X POST -d '{"dst-ip": "10.0.0.7/32", "dst-ip": "10.0.0.3/32", "nw-proto":"ICMP"}' http://localhost:8080/wm/firewall/rules/json
```

After executing the command, it shows prompt: Rule added.

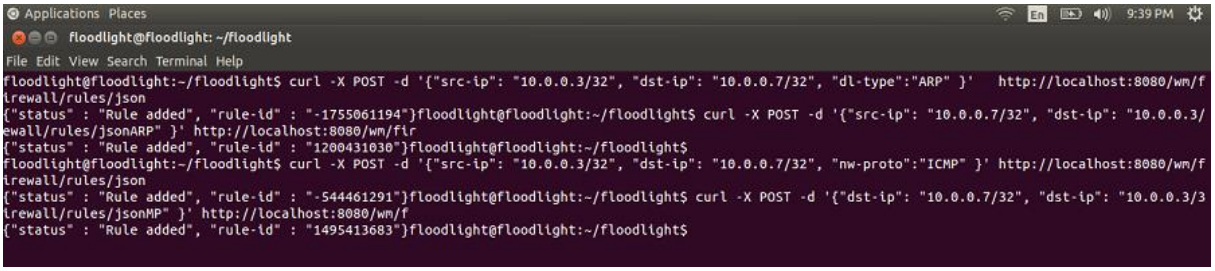

Fig.6: Adding ALLOW Rule between two IP Hosts

7. **<u>Adding an ALLOW rule for UDP (such as iperf) to work between IP hosts:</u>** To add ALLOW rule for UDP to work between two ip hosts, we used command `curl -X POST -d '{"src-ip": "10.0.0.4/32", "dst-ip": "10.0.0.10/32", "dl-type":"ARP" }' http://localhost:8080/wm/firewall/rules/json`

```
curl -X POST -d '{"dst-ip": "10.0.0.10/32", "dst-ip": "10.0.0.4/32", "dl-type":"ARP" }' http://localhost:8080/wm/firewall/rules/json

curl -X POST -d '{"src-ip": "10.0.0.4/32", "dst-ip": "10.0.0.10/32", "nw-proto":"UDP" }' http://localhost:8080/wm/firewall/rules/json

curl -X POST -d '{"src-ip": "10.0.0.10/32", "dst-ip": "10.0.0.4/32", "nw-proto":"UDP" }' http://localhost:8080/wm/firewall/rules/json
```

After executing the command, it shows prompt: Rule added.

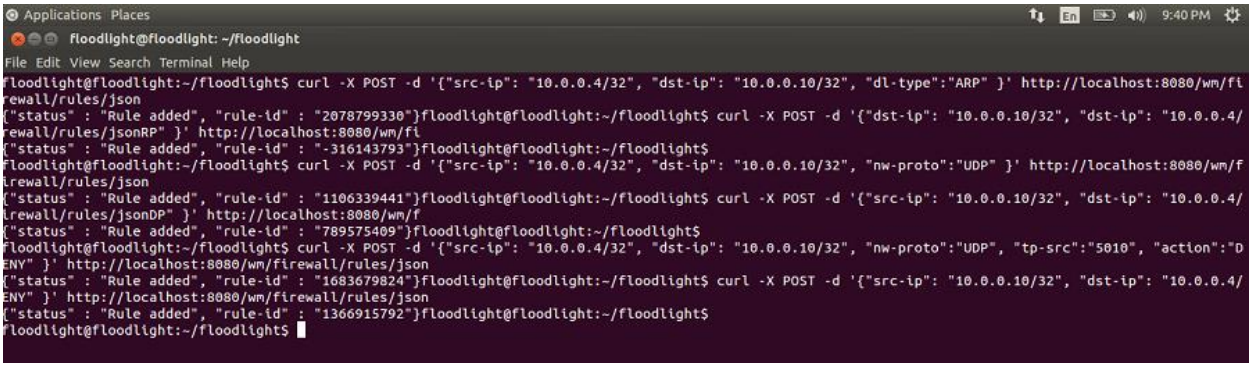

Fig.7: Adding ALLOW rule for UDP