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

CAN201

Dr. Wenjun Fan

Outline

- Prerequisite
- Steps for performing this lab
- Summary of the security problem
- Security solution thoughts
- Demo
- In-class Test 3

Prerequisite

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

Steps

Step1:

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** ”

```
root@bitcoinattacker:/home/wfan/Desktop# ryu-manager lab11.py
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

Step2:

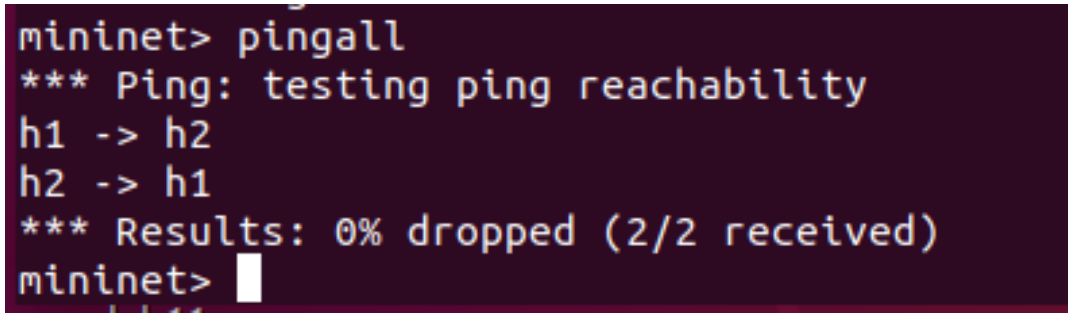
1) Open another new terminal
2) Run ‘ **sudo mn --controller=remote,ip=127.0.0.1,port=6653 --switch=ovsk,protocols=OpenFlow13** ’ to run a Mininet Topology

- 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

Step3:

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



Steps

Step4:

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

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
```

Steps

Step5:

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

Step6:

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
```

Steps

Step7:

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
```

Steps

Step8:

1) Wait ~2 minutes and repeat the previous Ping 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
```

Steps

Step9:

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
```

Summary of the security problem

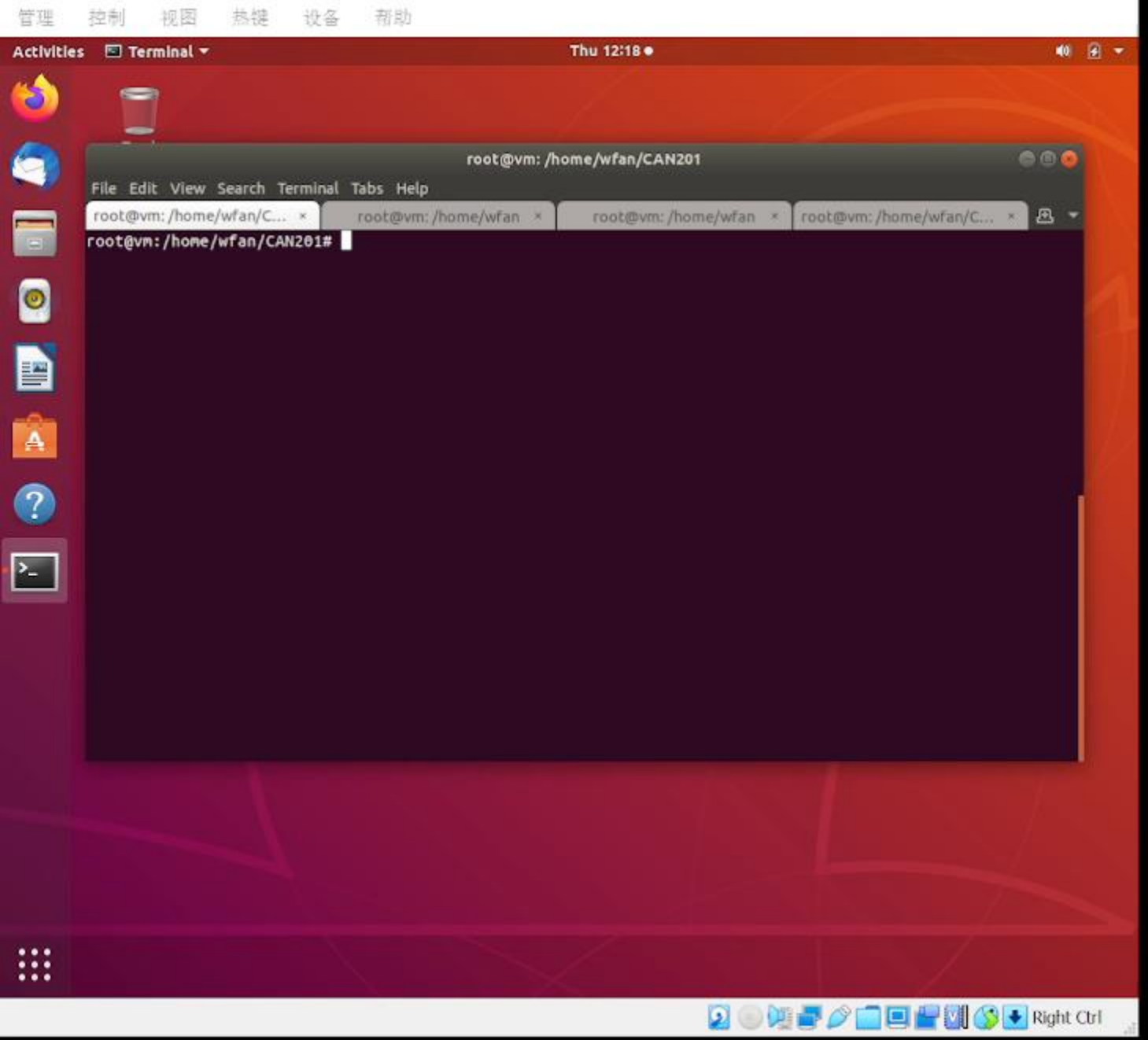
- This is a DoS attack in the data plane (the SDN-switch) of SDN.

- When the flow table of OVS switches is full, any additional flow-rule installation will be failed 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?
  - Controller application (lab11.py)!
- Why? (Answer for doing the Task)
- Can we avoid it?
  - Yes, see the Demo.

- Demo


管理 控制 视图 热键 设备 帮助
Activities
Terminal
Thu 12:18
root@vm: /home/wfan/CAN201
File Edit View Search Terminal Tabs Help
root@vm: /home/wfan/C...
root@vm: /home/wfan
root@vm: /home/wfan
root@vm: /home/wfan/C...
root@vm:/home/wfan/CAN201#
Right Ctrl

In-class Test 3

- Go through all the above instructions.

- Go to LMO to fetch the specification of In-class Test 3, take the test, and submit the answer to LMO.