Hands-on Lab Description

*CS-CNS-00103 –
DOS attack on SDN controller*

Category:

```
CS-CNS: Computer Network Security
```

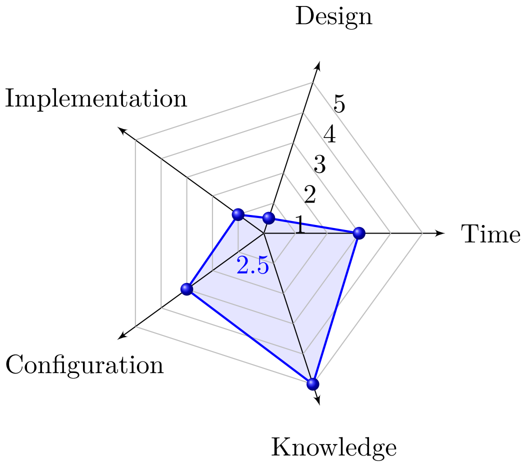
Objectives:

```
Understand Denial of Service attack
Setup POX controller
Setup mininet and create topology using mininet
Simulate DOS attack using POX controller and mininet
```

Estimated Lab Duration:

```
Expert: 50 minutes
Novice: 120 minutes
```

Difficulty Diagram:



Difficulty Table.

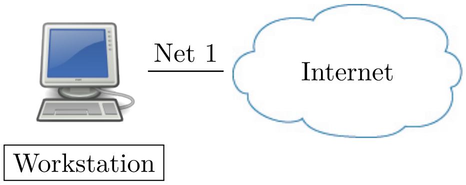
Measurements	Values (0-5)
Time	3
Design	0.5
Implementation	1
Configuration	3
Knowledge	5
Score (Average)	2.5

Required OS:

```
Linux: Ubuntu 18.04 LTS (Bionic Beaver)
```

Lab Running Environment:

```
VirtualBox https://www.virtualbox.org/ (Reference Labs: CS-SYS-00101)
```



```
Client: Linux (Ubuntu 18.04 LTS)
Client-side Net 1: 10.0.2.0/24
```

Lab Preparations:

```
Open vSwitch Installation and Basic Setup (CS-NET-00006)
Mininet Installation (CS-NET-00007)
Controller based lab for POX controller (CS-NET-00008)
Containernet Lab (CS-NET-00009)
SDN stateless firewall (CS-CNS-00101)
Required packages installed: mininet, POX, OVS
```

Lab Overview

In this lab you will emulate Denial of Service (DoS) attacks in an SDN networking environment. DDoS Attacks can target various components in the SDN infrastructure. You will need to set up an SDN-based firewall environment based on containernet, POX controller, and Over Virtual Switch (OVS). To mitigate DoS attacks, you need to develop a port security solution to counter the implemented DoS attacks.

The assessment of the lab is based on the completeness of implementing firewall filtering rules that satisfy the required firewall security policies. Students need to submit a sequence of screenshots and corresponding illustrations to demonstrate how they fulfilled the firewall's packet filtering requirements.

In summary, students will do:

- Setup the OpenFlow-based SDN environment using miniment or containernet.
- Setup POX OpenFlow controllers.
- Implement DoS attacks targeting the flow controller.
- Implement port security approaches to counter DoS attacks.
- Demonstrate the port-security based mitigation solution.

In the following tasks, task 1 guides you to set up the system environment; task 2 shows a step-by-step procedure to practice a simple DoS case study, and then and the lab uses a simple SDN firewall approach to mitigate the attack. By finishing the task 2, you will need to fulfill new requirements given in Task 3 to build new sets of rules to enable and disable network flows.

Task 1.0 Preparation of setting up lab environment

Suggestions for Task 1:

1. Review and exercise the the following labs: Open vSwitch and Basic Setup (CS-NET-00006), Mininet (CS-NET-00007), POX (CS-NET-00008), and*containernet* (CS-NET-00009). These labs help you understand SDN and exercise basic setup of an SDN environment.
2. Review and exercise the lab CS-CNS-00101 (SDN firewall). This lab will provide you basic knowledge foundation to run this lab on how to set up flow filtering rules.

Note that the services (Mininet, OVS and POX) may have already been setup on your server. You need to verify if these servers are setup properly to conduct your required tasks.

Before starting the lab, you need to check several software components and see if they have been set up properly. They are:

- Check if python is installed

```
$       python --version
```

- Check if python3.x is installed

```
$       python3 --version
```

- Check if *mininet* is installed (or check CS-NET-00007 for *mininet* installation and setup)

```
$       mn --version
$       sudo mn --test pingall
```

This will create a temporary hosts and switch and pings to all hosts that it created. If*mininet* is installed properly, it will execute and clean all the created hosts later and exit successfully.

- Check installation of pox. Go to directory where POX is installed, e.g. a common source code of POX is installed in directory */home/ubuntu/pox*. Here, we use $POX_DIR represents the POX director in your system.

```
$      cd $POX\_DIR % depends on where the pox folder is created.
$      ./pox.py -verbose forwarding.hub
```

- Check OVS installation

```
$      ovs-vsctl --version
```

It should display OVS version and details when it was installed, something like below:

```
     ovs-vsctl (Open vSwitch) 2.9.5
     DB Schema 7.15.1
```

- Check if wget is installed.

```
$      wget --version
```

If It is not installed, install it using following command.

```
$      sudo apt install wget
```

- Check if hping3 is installed.

```
$      hping3 --version
```

If It is not installed, install it using following command.

```
$      sudo apt-get install hping3
```

Task 2.0 Simulate DOS attack on SDN

In a Software Defined Networking (SDN) environment, hosts are connected to a controller via switches, in which the data flow is numbered in the Figure CS-CNS-00103.2 to show how its initial flow packets are processed. A new flow will be firstly intercepted by the controller, and then the controller will apply appropriate flow rules to enable the data flow from the source (e.g., host A) to the destination (e.g., host B).

Now, let's assume the attacker, i.e., host A, will try to send multiple spoofed packets to victim hosts, e.g., host B, sharing same switch so that it will block the switch and the controller by overloading the controller with multiple spoofed packets to prevent the controllers implement flow rules on SDN switches. This will potentially breaks down whole network.

Task 2.1 Getting source code

After you have verified that POX and *mininet* are installed, Open POX controller in one terminal window and run the controller forwarding application in that window.

1. Download CS-CNS-00103 source code for stateless firewall, if have not done before. (Pre-requisite: Check

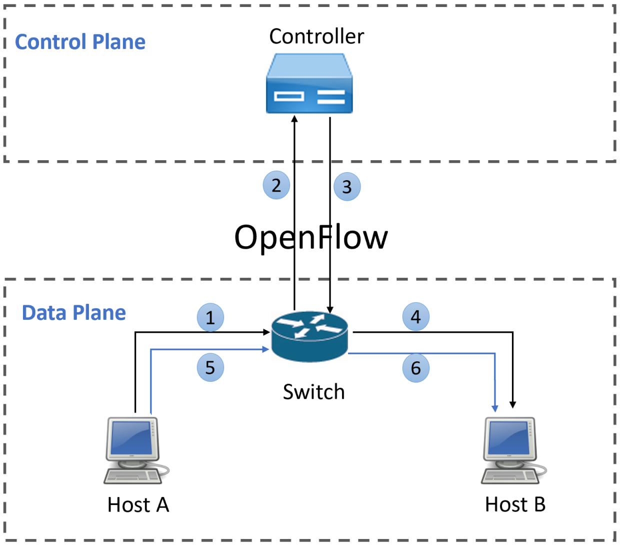


Figure CS-CNS-00103.2
SDN traffic flows.

if wget is installed using command *wget –version*. If wget is not installed, install it using *'sudo apt install wget'*)

```
$      wget https://gitlab.thothlab.org/thoth-group/ThoThLabResource/raw/master/
         lab-cs-cns-00103.zip
$      unzip lab-cs-cns-00103.zip
```

```
     Source code files are located in the folder ‘lab-cs-cns-00103’.
```

2. We need to use this firewall application with POX controller. To do so, copy *L3Firewall.py* file into the pox folder such as *./pox/pox/forwarding*. Here we assume POX_DIR is directory where POX source code is present, e.g., If you have installed POX in */home/ubuntu/*, then your $POX_DIR is */home/ubuntu/pox*.

```
$      sudo cp lab-cs-cns-00103/l2firewall.config $POX_DIR/. % copy the layer 2
         config file.
$      sudo cp lab-cs-cns-00103/l3firewall.config $POX_DIR/. % copy the layer 3
         config file.
$      sudo cp lab-cs-cns-00103/L3Firewall.py $POX_DIR/pox/forwarding/. % copy the
         firewall application file.
```

3. Run *mininet* using *containernet*. You need to start *mininet* configuration with 9 *containernet* hosts, one remote controller which will be connected to POX controller and one ovs switch.

   Now, you can open a new terminal window, run the following command:

```
$      sudo mn --topo=single,9 --controller=remote,port=6655 --switch=ovsk --mac
```

   It will create a *mininet* environment in *containernet* with 9 *containernet* hosts, one OVS switch and one remote controller. Option –mac will assign small, unique and fixed set of mac address based on host id. It will remain constant after every run. The topology should look like topology is shown in Figure CS-CNS-00103.3. In this setup, it open the controller at port 6655. Note that the default POX controller port number is 6633, and you can open multiple ports with multiple controllers (how to do it?).

The output will be something like the below:

```
root@ubuntu:~# sudo mn --topo=single,9 --controller=remote,port=6655
    --switch=ovsk --mac
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9,
    s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Starting controller
c0 c1
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet>
```

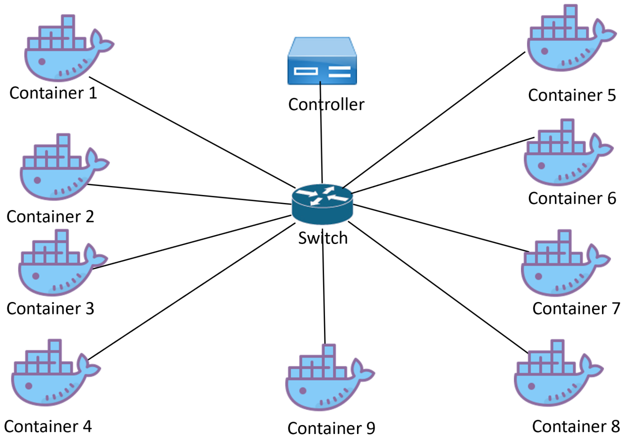


Figure CS-CNS-00103.3
Network topology for mininet-based SDN environment.

4. Run POX controller, where here we assume *$POX_DIR* is the POX directory where pox source codes present:

```
$      cd $POX_DIR % It is where your pox file is located, a common location is at
         /home/ubuntu/pox
$      sudo ./pox.py openflow.of_01 --port=6655 pox.forwarding.l3_learning
```

Here POX controller is invoked by *./pox.py* command and we run *l3_learning.py* application and *L3Firewall.py* file from POX controller. POX uses the convention to run applications such as "pox.subdirectory.filename", and the applications given by this way has to run using *./pox.py* with *pox.subdirectory.filename* as an argument. Running the pox command allow POX listen to the localhost *0.0.0.0:6633* for switch request, where 6633 is the default port for POX. Running on a different port, you can use option such as *–port=6655* as an example. The output of this command is shown as follows:

```
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
```

Using POX, all the forwarding applications has to be stored in */pox/pox/forwarding* directory. To give relative path from directory where POX binary is present, we follow convention *pox.forwarding.L3Firewall*

Following table CS-CNS-00103.2 depicts the mapping between MAC addresses and respective IP addresses in specified *mininet* topology. Since we have enabled *–mac* option in *mininet* topology, MAC address fill be fixed for each *containernet* host in every run.

Container Host	**Layer 2 Address (MAC address)**	**Layer 3 Address (IP address)**
Container host h1	00:00:00:00:00:01	10.0.0.1
Container host h2	00:00:00:00:00:02	10.0.0.2
Container host h3	00:00:00:00:00:03	10.0.0.3
Container host h4	00:00:00:00:00:04	10.0.0.4
Container host h5	00:00:00:00:00:05	10.0.0.5
Container host h6	00:00:00:00:00:06	10.0.0.6
Container host h7	00:00:00:00:00:07	10.0.0.7
Container host h8	00:00:00:00:00:08	10.0.0.8
Container host h9	00:00:00:00:00:09	10.0.0.9

Table CS-CNS-00103.2
Mapping of MAC addresses with IP addresses in respective *containernet* hosts

Note that in the following subsections, you can use two ways to test your virtual networks from the command-line of a host:

(a) You can specify from which host to initiate a command. For example:

```
        containernet> h1 ping h3  % ping from h1 to h3
        containernet> h1 hping3 -c 5 h2 -V --tcp-timestamp % use hping3 to
            sent tcp packets to h2
```

(b) you can start a host terminal and use network configuration to perform the test. For example, the following command is to start an x-terminal of host h1:

```
        containernet> xterm h1
```

Once the x-terminal is started you can run the above commands just like in a real host command-line, such as:

```
$         ping 10.0.0.3   % ping from h1 to h3 (10.0.0.3)
$         hping3 -c 5 10.0.0.2 -V --tcp-timestamp % use hping3 to sent tcp
            packets to h2 (10.0.0.2)
```

Task 2.2 Simulate DDoS attack

Using the mininet topology created in above steps, now simulate a DoS attack in SDN controller.

1. From *containernet* environment, Open host *h1* console. This will open xterm window for host *h1* from the *mininet* environment.

```
$       containernet> xterm h1
```

2. Check initial Openvswitch flow entries. In this lab, Open vSwitch is used. The new rules added based on the match and controller signals, can be viewed from OVS switches. To do so, you have to dump flow entries at OVS switches and use OpenvSwitch commands to verify and check the rules getting added in the Switches.

   When you run *l3_learning* application from POX and create a *mininet* environment, an SDN is formed with *mininet* containers as hosts, OVS as a data plane switch to handle switching functionalities and POX to handle control plane functionality.

   Open another terminal to run OVS commands that will verify the functionality of POX.

   At this moment, since the network has just started, there are no flows and packet stats available in the ovs-switch S1.

   Now, open another terminal window and run the following command:

   ```
   $       ovs-ofctl dump-flows s1
   ```

3. Start flooding from host *h1* to *h2*.

   From the x-terminal window, send spoofed tcp syn-flood packets from *h1* to *h2* with random source IP to the *h2* node:

   ```
   $       hping3 10.0.0.2 -c 10000 -S --flood --rand-source -V
   ```

4. In a separate terminal, check openvswitch flow entries

   ```
   $       ovs-ofctl dump-flows s1
   ```

   Verify if we have successfully attempted DOS attack on SDN controller. The output may be something like the follow, in which it shows four continuously captured packets:

   ```
        cookie=0x0, duration=0.066s, table=0, n_packets=0, n_bytes=0,
            idle_timeout=10,priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,
            dl_src=00:00:00:00:00:01,
            dl_dst=00:00:00:00:00:02,nw_src=16.66.66.28,nw_dst=10.0.0.2,nw_tos=0,
            tp_src=58949,tp_dst=0
            actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
        cookie=0x0, duration=0.066s, table=0, n_packets=0, n_bytes=0,
            idle_timeout=10,priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,
            dl_src=00:00:00:00:00:01,
            dl_dst=00:00:00:00:00:02,nw_src=81.24.27.139,nw_dst=10.0.0.2,nw_tos=0,
            tp_src=59448,tp_dst=0
            actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
        cookie=0x0, duration=0.066s, table=0, n_packets=0, n_bytes=0,
            idle_timeout=10,priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,
            dl_src=00:00:00:00:00:01,
            dl_dst=00:00:00:00:00:02,nw_src=99.79.229.144,nw_dst=10.0.0.2,nw_tos=0,
            p_src=59449,tp_dst=0 actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
        cookie=0x0, duration=0.066s, table=0, n_packets=0, n_bytes=0,
            idle_timeout=10,priority=65535,tcp,in_port="s1-eth1",vlan_tci=0x0000,
            dl_src=00:00:00:00:00:01,
            dl_dst=00:00:00:00:00:02,nw_src=150.151.121.171,nw_dst=10.0.0.2,nw_tos=0,
            tp_src=59450,tp_dst=0
            actions=mod_dl_dst:00:00:00:00:00:02,output:"s1-eth2"
   ```

   In this example, you can observe that there are four different source IP addresses (*nw_src*):

16.66.66.28, 81.24.27.139, 99.79.229.144, and 150.151.121.171 transmitted from the same MAC address "*dl_src*=00:00:00:00:00:01". It is a strong indication that *h1* is spoofing other nodes to generate excessive amount of network traffic.

Now ping from host *h4* to host *h9* from same *containernet* environment.

```
$       containernet> h4 ping h9
```

We should be able to see ICMP "destination Host Unreachable" response for some time. This demonstrates that the SDN controller was flooded and cannot respond for any new incoming new flows.

Now, you can stop the flooding by killing the hping3 using *Ctrl+C* in *h1*'s x-terminal. After waiting for few more seconds, the service will resume and we could see valid ping responses from *h9* back to *h4*. This shows, the controller is resumed back to work normally.

Task 2.3 Mitigate DoS Attack by implementing port security

Now, you can apply SDN switch rules to mitigate DoS Attacks.

1. How to setup SDN-based firewall is described in lab CS-CNS-00101. In Task 2.2, we can use the following command to show multiple flows originated from source mac address *00:00:00:00:00:01* having different source IP addresses that target at *h2* (*10.0.0.2*). Thus, a straightforward firewall rule can be setup is to block the sender **host h1** which is attacking the system with DoS attacks.

   To block a host matching source MAC as *00:00:00:00:00:01*, we can add a firewall rule in *l2firewall.config* file.

   The input configuration file *l2firewall.config* for firewall application contains following rule fields for each line in the configuration file:

   ```
   priority, source MAC, destination MAC
   ```

   Please make sure that *l2firewall.config* file should contain following rule:

   ```
   id,mac_0,mac_1
   1,00:00:00:00:00:01,00:00:00:00:00:02
   ```

   Note that POX takes the default IP ranges from "10" network if no specified IP address is given for each host, in which it assigns 10.0.0.1 to *h1* . . . 10.0.0.9 to *h9*. This firewall configuration file consists of firewall rules for *BLOCKING* traffic. In *l2firewall.config* file, the first rule BLOCKS packets from source MAC address 00:00:00:00:00:01 to destination MAC address 00:00:00:00:00:02. Stop any other POX instance by pressing *Ctrl + c* in POX window.

   Run POX controller with firewall application this time, where we assume *$POX_DIR* is the POX directory where pox source codes present:

   ```
   $       cd $POX_DIR % It is where your pox file is located, a common location is at
             /home/ubuntu/pox
   $       sudo ./pox.py openflow.of_01 --port=6655 pox.forwarding.l3_learning
             pox.forwarding.L3Firewall
   ```

   Here POX controller is invoked by *./pox.py* command and we run *l3_learning.py* application and *L3Firewall.py* file from POX controller. All the forwarding applications has to be stored in */pox/pox/forwarding* directory. To give relative path from directory where POX binary is present, we follow convention *pox.forwarding.L3Firewall*. The *L3Firewall.py* will invoke *l2firewall.config* and *l3firewall.config* to add blocking rules at layer 2 and layer 3, respectively. Please refer to lab CS-CNS-00101 for more details on how to setup a firewall blacklist rule sets at layer 2 and layer 3.

2. Now, you can verify if the flooding attack to DNS controller will work or not by starting flooding again from host *h1* to *h2*.

   From the x-terminal window, send spoofed tcp syn-flood packets from *h1* to *h2* with random source IP to the *h2* node

   ```
   $      hping3 10.0.0.2 -c 10000 -S --flood --rand-source -V
   ```

3. In a different command-line terminal, check openvswitch flow entries.

   ```
   $      ovs-ofctl dump-flows s1
   ```

4. You can verify if the switch has successfully blocked DOS attack targeting at the POX controller. You can check if you can ping from any hosts except *h1*:

   ```
   $      containernet> h4 ping h9
   ```

   You should be able to see normal ICMP response.

   You can also see corresponding OpenFlow entries in the ovs stats. To check this, run following command in another terminal:

   ```
   $      ovs-ofctl dump-flows s1
   ```

Task 3 Requirements for performing DoS attack and Blocking DoS attack

In this lab, please perform DoS attack in SDN environment using POX SDN controller and *mininet* emulated environment. Understand what DoS attack is and try to attack any host *h1*- *h4*. Later on, you have to stop the DoS attack by applying port security on the SDN network based on setting up new flow rules to block the DoS attack source. The basic approach is called *Port Security*.

Port security is a layer two traffic control feature originally implemented on Cisco Catalyst switches. It enables an administrator configure individual switch ports to allow only a specified number of source MAC addresses ingressing the port. Its primary use is to deter the addition by users of “dumb” switches to illegally extend the reach of the network (e.g., so that two or three users can share a single access port). In this lab, we meant port security is to limit a MAC address to be assigned only to one IP address. The pseudo code of the to-be implemented port security is presented as follows:

```
Initiate a port table (PT);
For any newly received flow F originated from the source MAC address F.SrcMAC;
   if F.SrcIP is new;
      update PT with the mapping F.SrcMAC <--> F.SrcIP;
   else
      block F.SrcMAC % Block an MAC address that had spoofed multiple IP addresses
end
```

Before implementing the described port security, you need to fulfill the following requirements.

1. Create a *mininet* topology with following components:

   - *containernet* host *h1*
   - *containernet* host *h2*
   - *containernet* host *h3*

- *containernet* host *h4*
- OVS switch *s1*
- remote controller *c1*.

Set the following links:

- *container* host *h1* to switch *s1*
- *container* host *h2* to switch *s1*
- *container* host *h3* to switch *s1*
- *container* host *h4* to switch *s1*
- controller *c1* to switch *s1*

It should be single topology with above specification.

2. Set IP addresses of *containernet* hosts with following specifications.

   - Set IP address 192.168.2.10 to *container* host #1 interface h1-eth0
   - Set IP address 192.168.2.20 to *container* host #2 interface h2-eth0
   - Set IP address 192.168.2.30 to *container* host #3 interface h3-eth0
   - Set IP address 192.168.2.40 to *container* host #4 interface h4-eth0

3. Assume the attacker generates DoS attack packets from *h1*, and it can choose any targeting host to deploy the DoS attack as described in Task 2.

4. Implement the port security described in the above pseudo code. The requirements is given as follow:

   - You can use any program languages that you are familiar with to automate the DoS detection and flow rule setup. Since the POX controller is implemented in Python, thus Python would be the best option for you to implement the desired pseudo codes.
   - There are basically two approaches to implement the pseudo codes: (a) write a program to monitor the out put of "ovs-ofctl dump-flows s1", which is presented in task 2.2. Identify if you have seen multiple IP address generated from the same MAC address and update the new l2firewall.config, and reapply the new flow rules. (b) write the port security function in your L3Firewall.py directly.

Bonus Points: The provide pseudo codes do not fully consider sophisticated DoS attacks. For example, the attacker change its source MAC address to make it looked generated from different MAC addresses. Can you design a solution and demonstrate how it works? The bonus points will be considered a max of 10% of your project grade depending on the completeness of your solution and implementation.

Deliverable

Students need to submit a sequence of screenshots with explanations on which they can achieve requirements described in the *Lab Assessment* section. Moreover, students is required to submit the developed python codes for evaluation. On how to submit screenshots and codes/file please refer to the *Guidance for Students* Section B.4 (Submit Lab Reports).

Related Information and Resource

```
POX Github:  https://noxrepo.github.io/pox-doc/html/
POX Controller Tutorial: http://sdnhub.org/tutorials/pox/
Open vSwitch Cheat Sheet: https://therandomsecurityguy.com/openvswitch-cheat-sheet/
Containernet: https://containernet.github.io/
Containernet tutorial: https://github.com/containernet/containernet/wiki/Tutorial:-Getting-Started
Port security: https://packetlife.net/blog/2010/may/3/port-security/
```

Other frequently used OVS command

1. Getting general information

```
$       ovs-vsctl list open_vswitch
$       ovs-vswitchd -V
```

2. Check the status of *openvswitch*

```
$       service openvswitch status
```

3. List all bridges

```
$       ovs-vsctl list bridge
```

4. List All ports

```
$       ovs-vsctl list port
```

5. Show mac learning table for a bridge

```
$       ovs-appctl fdb/show br0
```

6. Show flows on *ovs*

```
$       ovs-dpctl show -s
```

7. Dump flows

```
$       ovs-dpctl dump-flows br0
```

8. To list bonds

```
$       ovs-appctl bond/list
```

9. To show bond properties for a bond

```
$       ovs-appctl bond/show bond0
```

Lab Assessment (100 points)

In this lab, the users is required to simulate DOS attack on SDN controller using POX controller and *containernet* or *mininet* to simulate the network in simulated container environment.

1. (15 points) Setting up *mininet* and Running *mininet* topology
   (a) Create a *mininet* based topology with 4 container hosts and one controller switches.
   - Add link from controller to switch 1.
   - Add link from switch 1 to container 1.
   - Add link from switch 1 to container 2.
   - Add link from switch 1 to container 3.
   - Add link from switch 1 to container 4.

   (b) Run the *mininet* topology. (Create *mininet* topology using *mininet* command-line)

2. (15 points) Should assign IP addresses to hosts.

   Make the interfaces up and assign IP addresses to interfaces of *container* hosts:
   Assign IP address 192.168.2.10 to *container* host #1
   Assign IP address 192.168.2.20 to *container* host #2
   Assign IP address 192.168.2.30 to *container* host #3
   Assign IP address 192.168.2.40 to *container* host #4

3. (15 points) Perform Flood attack on SDN controller following a suggested procedure:
   (a) Run *l3_learning* application in POX controller.
   (b) Check openflow flow-entries on switch 1.
   (c) Start flooding from any container host to container host #2. using source address 10.10.10.1
   (d) Check Openflow flow entries at switch 1

4. (55 points) Mitigate DoS attack by implementing port security and using OpenFlow based firewall.
   - (25 points) You should illustrate (through screenshots and descriptions) your implemented program codes.
   - (15 points) You should demo how your implementation can mitigate the DoS through a sequence of screenshots with explanation.
   - (15 points) You should submit the source codes of your implementation.