Ira A. Fulton Schools of Engineering
Arizona State University

CSE 548: Advanced Computer Network Security

SDN-Based DoS Attacks and Mitigation Project

Purpose

This project offers hands-on experience in understanding, detecting, and mitigating denial-of-service (DoS) attacks in a SDN network environment. Learners can simulate DoS attack to grasp their impact on network resources and services. Through this project, you will implement mitigation techniques like traffic filtering, rate limiting, and deploying Openflow flow rules to stop the attack. By engaging with real-world scenarios, learners develop practical skills in defending networks against DoS attacks, enhancing their understanding of network security concepts and strategies.

Objectives

Learners will be able to:

- Understand Denial of Service attack
- Setup Openflow environment, flow rules, and do packet processing using Openflow
- Setup a simple Open Virtual Switch (OVS) network using miniment or containernet
- Setup POX OpenFlow controllers
- Simulate DOS attack using POX controller and mininet
- Implement flow-based filtering rules based on flow control requirements.

Technology Requirements

- Hardware
    - Intel or AMD based computer with 8GB or more memory.
    - NOTE: 6GB is technically sufficient, but performance will be sluggish.

- Software
    - VirtualBox 5.0 or Newer
    - Windows 10, Mac OS X, or Linux 64-bit as the base operating system.
    - Access to Apporto through a web browser

- Linux: Ubuntu 18.04 LTS (Bionic Beaver)

Project Description

In this project you will emulate Denial of Service (DoS) attacks in an SDN networking environment. 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 will need to develop a mitigation strategy to counter the implemented DoS attacks.

Estimated time to complete: 8 hours

Specific tasks:

- Set up the OpenFlow-based SDN environment using miniment or containernet.
- Set up 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.

Directions

You may work on your project locally or through Apporto. Follow the access guidelines provided and then review the Lab Preparation steps to complete your work. Headings or titles labeled with “Local only” are specific to the local setup, otherwise you may complete your work through Apporto as well. Please make sure your final deliverable follows the requirements.

Project Files

The project files include the VM image file, the assigned project, and associated background labs. These can be found in your course on the Project Overview page.

Local Setup

This project uses the virtual machine image file that is same as **SDN-Based Stateless Firewall Project**. If this was set-up in the previous projects you do not need to set-up the environment a second time. You can use the same VM for this project.

Accessing Apporto

For this project, you will work through Apporto’s modular cyberlabs available through the course. Review all the project directions before starting so you know how to submit your work correctly:

1. In the course, click “Apporto Virtual Lab – App Store” (be patient while the lab loads)
2. In the App Store, select the “**CSE 548 Modular CyberLab**” lab. Please be patient as the lab initializes, which may take a few moments.
3. Once the lab environment is ready, you will be on a Linux machine and you need to click on the “**GNS3**” app from the Activities Menu.
4. In the pop-up window, choose “Open a project from disk”. Navigate to the path Home → Labs → projects → CSE548 Project 3 and then select the “**CSE548 Project 3.gns3**” file.
5. You will see two (2) components listed: a Linux Server(Project3) and a NAT Network(NAT1) with an IP range of 10.0.2.0/24.
6. Right-click on the Linux server and select “**Start”**, to start the server. You will know it’s running when the red block next to the server changes to green.
7. Again, right-click on the server and select “**console**” to access the Linux desktop.
8. Server credentials:
    a. Username: ubuntu
    b. Password: 123456
9. You are all set! You can now start working on your project.

Note*: Learners should refrain from using the virtual machine (VM) for personal purposes. This VM is provided solely for academic and course-related activities. Any personal activities may interfere with its intended purpose and impact the learning environment for yourself and your peers.*

Lab Preparation

Background Labs

- 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

Task 1.0 Preparation of setting up the lab environment

Suggestions for Task 1:

1. Review and exercise 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 the 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 set up on your server. You need to verify if these servers are set up 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 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 to represent 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 the 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 the following command:

  ```
  $ sudo apt install wget
  ```

- Check if hping3 is installed.

  ```
  $ hping3 --version
  ```

  If It is not installed, install it using the 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 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 the 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 break down the whole network.

Task 2.1 Getting source code

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

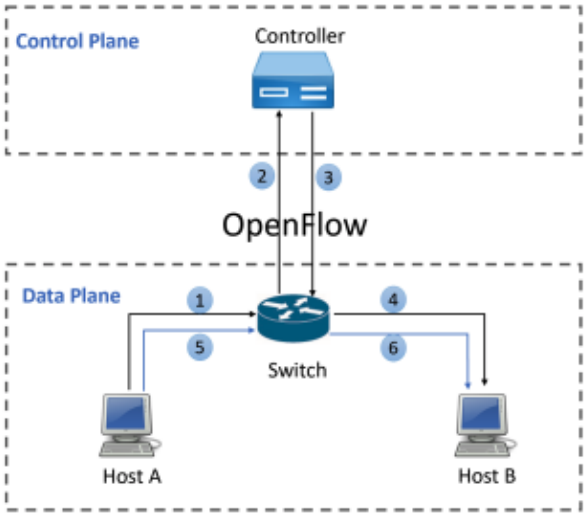

Figure CS-CNS-00103.2
SDN traffic flows

1. Download the CS-CNS-00103 source code for a stateless firewall, if have not done before. (Pre-requisite: Check if wget is installed using the command wget –version. If wget is not installed, install it using 'sudo apt install wget')

```
$ wget
https://github.com/SaburH/CSE548/raw/main/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 a POX controller. To do so, copy the L3Firewall.py file into the pox folder such as ./pox/pox/forwarding. Here we assume POX DIR is the 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 the mininet configuration with 9 containernet hosts, one remote controller which will be connected to the POX controller, and one OVS switch. Now, you can open a new terminal window, and run the following command:

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

It will create a mininet environment in a containernet with 9 containernet hosts, one OVS switch, and one remote controller. Option –mac will assign a small, unique, and fixed set of Mac addresses based on host ID.

It will remain constant after every run. The topology should look like the topology shown in Figure CS- CNS-00103.3. In this setup, it opens 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>
```

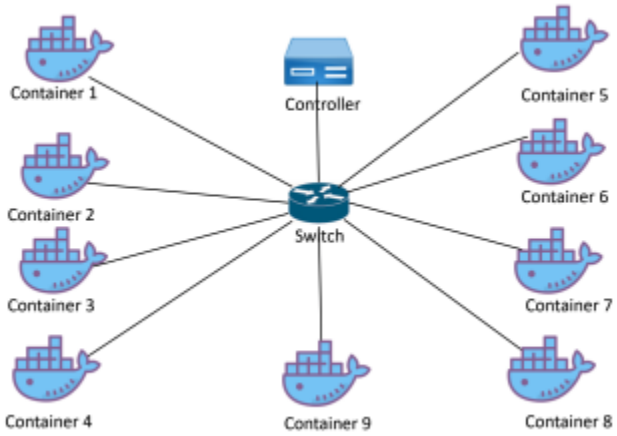

Figure CS-CNS-00103.3
Network topology for mininet-based SDN environment.

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 the POX controller. POX uses the convention to run applications such as “pox.subdirectory.filename”, and the applications given this way have to run using ./pox.py with pox.subdirectory.filename as an argument. Running the pox command allows POX to listen to the localhost 0.0.0.0:6633 for switch requests, where 6633 is the default port for POX. Running on a different port, you can use options 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 have to be stored in /pox/pox/forwarding directory. To give a relative path from the directory where the POX binary is present, we follow convention pox.forwarding.L3Firewall. The 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, the MAC address will 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 the 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 an excessive amount of network traffic.

Now ping from host h4 to host h9 from the 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 to any new incoming flows.

Now, you can stop the flooding by killing the hping3 using Ctrl+C in h1’s x-terminal. After waiting for a few more seconds, the service will resume and we can 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 set up an SDN-based firewall is described in lab CS-CNS-00101. In Task 2.2, we can use the following command to show multiple flows originating 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 set up 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 the firewall application contains the following rule fields for each line in the configuration file:

priority, source MAC, destination MAC

Please make sure that the l2firewall.config file contains the 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 the 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 the 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 the l3 learning.py application and L3Firewall.py file from the POX controller. All the forwarding applications have to be stored in /pox/pox/- forwarding directory. To give a relative path from the directory where the 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 set up a firewall blacklist rule sets at layer 2 and layer 3.

2. Now, you can verify if the flooding attack to the 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 the DOS attack targeting 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 the following command in another terminal:

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

Task 3.0 Requirements for performing DoS attack and Blocking DoS attack

In this lab, please perform a DoS attack in the SDN environment using the POX SDN controller and mininet emulated environment. Understand what a 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 to 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 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 a 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 the 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 a single topology with the above specification.

2. Set IP addresses of containernet hosts with the 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 are given as follows:
    - You can use any programming language 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 output of “ovs-ofctl dump-flows s1”, which is presented in task 2.2. Identify if you have seen multiple IP addresses 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 provided pseudo codes do not fully consider sophisticated DoS attacks. For example, the attacker changed its source MAC address to make it look generated from a different MAC address. Can you design a solution and demonstrate how it works? The bonus points will be considered a maximum of 10% of your project grade depending on the completeness of your solution and implementation.

Submission Directions for Project Deliverables

To complete the project, you will submit a sequence of screenshots and corresponding illustrations to demonstrate how they fulfilled the firewall's packet filtering requirements.This will be placed in the Proejct Report template provided on the Project Overview page in your course. Your work will be evaluated based on the completeness of implementing firewall filtering rules to demonstrate how to successfully counter the implemented DoS attack.

You are given an unlimited number of attempts to submit your best work. The number of attempts is given to anticipate any submission errors you may have in regards to properly submitting your best work within the deadline (e.g., accidentally submitting the wrong paper). It is not meant for you to receive multiple rounds of feedback and then one (1) final submission. Only your most recent submission will be assessed.

You must submit your SDN-Based DoS Attacks and Mitigation Project deliverable in this submission space. Learners may not email or use other means to submit any assignment or project for review, including feedback, and grading.

The SDN-Based DoS Attacks and Mitigation Project includes one (1) deliverable:

- **Project Report:** Use the Project Report template to submit your report as a DOC or PDF. Your file should be named using the following format: [Your Name_CSE 548_Name of Project or Assignment].
    - If you want to attach your l2 and l3firewall config files, please create a single .zip file that contains both the project report and all the other files). Follow the directions below to submit both zip files with your submission.

Report Submission

1. Navigate to your course and click on the "**Submission: SDN-Based DoS Attacks and Mitigation**" submission space
2. Click "**Start Assignment**" in the upper right corner
3. Click "**Upload File**" and add your completed project report template.
    a. To add another file, select, "**+ Add Another File**" and add additional files

4. When finished, click “**Submit Assignment**”
5. If needed: to resubmit the assignment
    a. Click “**New Attempt**” and add your file submission again
    b. When finished, click “**Submit Assignment**”

Evaluation

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

1. Setting up mininet and Running mininet topology
    a. Create a mininet-based topology with 4 container hosts and one controller switch.
        - Add a link from the controller to switch 1.
        - Add a link from switch 1 to container 1.
        - Add a link from switch 1 to container 2.
        - Add a link from switch 1 to container 3.
        - Add a link from switch 1 to container 4.
    b. Run the mininet topology. (Create mininet topology using mininet command-line)

2. Should assign IP addresses to hosts.
    a. 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. Perform a Flood attack on the 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. Mitigate DoS attacks by implementing port security and using OpenFlow-based firewall.

    - You should illustrate (through screenshots and descriptions) your implemented program codes.
    - You should demo how your implementation can mitigate the DoS through a sequence of screenshots with an explanation.
    - You should submit the source codes of your implementation.