Man-in-the-middle Attacks in the SDN Data Plane
Contents
Objective............................................................................................................................................... 3
Submission........................................................................................................................................... 3
Prepare the Experiment ..................................................................................................................... 3
  Prepare the Controller ..................................................................................................................... 3
  Prepare the Nodes .......................................................................................................................... 8
Conduct the Experiment ................................................................................................................... 12
  Scenario 1 (No MITM Attack) ..................................................................................................... 12
  Scenario 2 (MITM Attack) .......................................................................................................... 16
Objective

By using security protocols in the transport layer, students will learn how to establish secure communication between the controller and switch. Students will understand OpenFlow protocol vulnerability. Students will be able to launch the man-in-the-middle attack in SDN and understand how attackers can steal information. They will learn security protocols like TLS, IPSec, and SSH and their usage between the controller and the switches. They will learn authentication methods needed for all devices connected to the controller or switches to ensure secure communication.

Submission

Copy and modify the profile, then start the experiment using your modified profile. Get screenshots of the following two scenarios.

1) The **Relay node** did not conduct the MITM attack. Use TCPDUMP to capture packets in **Switch6** or **Switch4**

2) The **Relay node** conducted the MITM attack. Use TCPDUMP to capture packets in the **Relay node**.

CloudLab Login page: [https://www.cloudlab.us/login.php](https://www.cloudlab.us/login.php)

Prepare the Experiment

Prepare the Controller.

![Select a Profile](image)

Figure 1: Start an experiment using the **MITMController** profile. Then Click “Select Profile” button to proceed.
Figure 2: Click “Next” button to proceed.

Figure 3: Choose a cluster and click “Next” to proceed.
Figure 4: Click “Finish” to proceed.

Figure 5: The controller is successfully running.
Figure 6: Start a shell. Type **“ifconfig”** to check the IP. This IP will be used when we configure the switches. Let’s note it as **controller_ip**

<table>
<thead>
<tr>
<th>Topology View</th>
<th>List View</th>
<th>Manifest</th>
<th>Graphs</th>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>controller</strong>: ifconfig</td>
<td>ethernet</td>
<td>Link encapsulation</td>
<td>haddr: 02:28:df:cc:12:26</td>
<td>inet addr: fe80::28:ff:fecc:1226/64</td>
</tr>
<tr>
<td>lo</td>
<td>Loopback</td>
<td>inet addr: 127.0.0.1</td>
<td>Mask: 255.0.0.0</td>
<td>inet6 addr: ::1/128</td>
</tr>
</tbody>
</table>

Figure 7: setup controller.
1) Switch to root. Type **“sudo su -”**
2) Type **“wget https://people.cs.clemson.edu/~hongdal/set_floodlight.sh”**
Figure 8: Run the scripts.
1) Type “`chmod +x set_floodlight.sh`”
2) Type “`./set_floodlight.sh`”

Figure 9: Launch the controller.
1) Type “`cd floodlight`”
2) Type “`java -jar target/floodlight.jar`”
Prepare the Nodes

Download a topology from here: https://people.cs.clemson.edu/~hongdal/mitm-attack.xml
Create a new profile using the download topology.

Figure 10: Upload the topology when creating the profile.

Figure 11: Click “Edit topology” to configure the SDN controller.
Figure 12: Update the IP address to the **controller_ip**. Then click “Accept”.
Figure 13: Click “instantiate” to proceed.

Figure 14: Choose a cluster. Click “Next” to proceed.
Figure 15: The nodes are running successfully.
Conduct the Experiment

Scenario 1 (Normal Traffic)

1) Open the shells for each of the node. Do steps 2) and 3) for each node except for the Relay Node. We will use User1 as and example. The node IP should be different for different nodes.

2) Download the “set_ovs.sh” script from [https://people.cs.clemson.edu/~hongdal/set_ovs.sh](https://people.cs.clemson.edu/~hongdal/set_ovs.sh)

3) change permission and run the script.

```
"chmod +x set_ovs.sh"
"./set_ovs.sh eth1 eth2 155.98.37.66 10.10.10.1"
```

The red IP is the controller_ip, the blue IP is the node IP. Use different node IP for different nodes. E.g., 10.10.10.2 for User2, 10.10.10.3 for Switch4, and 10.10.10.4 for Switch6.

Figure 16: Download the script

Figure 17: Change permission and run
4) Check IP of **User1** and **User2**. Note that we will work with experimental IP (ovs-lan), not the public IP.

2) **PING** from **User1** to **User2** again.

![Network Topology and Interfaces](image)

**Figure 18:** Check the experimental IP of User1.
Figure 19: Check the experimental IP of User2.

3) Observe the packets from Switch4 or Switch6. Type “`tcpdump -i eth1 -nq icmp`” in their shells.

Figure 20: PINF from User1 to User2.

4) Observe the packets from Relay node. Type “`tcpdump -i eth1 -nq icmp`” in its shell.
Figure 21: Observe the packets at Switch4 or Switch6.
Scenario 2 (MITM Attack)

1) Stop the PING at User1
2) Conduct attack from Relay-Node.
   (1) Go to the Relay-Node shell.
   (2) Download the attack script from https://people.cs.clemson.edu/~hongdal/attack.sh
   (3) Type “chmod +x attack.sh”
   (4) Run attack.sh. “attack.sh eth1 eth2 155.98.37.66 10.10.10.5”. The controller_ip is use here.

Figure 22: Download attack.sh, change permission, and run the attack.sh script.

5) Wait for a minute to let the attack take effect.
6) PING from User1 to User2 again.
7) Observe the packets from the Relay-Node.
8) Observe the packets from Switch6 or Switch4.
Figure 23: Packets observed from Relay Node

```
root@relay-node:~# tcdump -i eth1 -nq icmp
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
21:57:28.856157 IP 10.10.1.1 > 10.10.1.2: ICMP echo request, id 2760, seq 1, length 64
21:57:28.867933 IP 10.10.1.1 > 10.10.1.2: ICMP echo reply, id 2760, seq 1, length 64
21:57:29.852329 IP 10.10.1.1 > 10.10.1.2: ICMP echo request, id 2760, seq 2, length 64
21:57:29.852883 IP 10.10.1.1 > 10.10.1.2: ICMP echo reply, id 2760, seq 2, length 64
21:57:30.829322 IP 10.10.1.1 > 10.10.1.2: ICMP echo request, id 2760, seq 3, length 64
21:57:30.853406 IP 10.10.1.1 > 10.10.1.2: ICMP echo reply, id 2760, seq 3, length 64
21:57:31.854138 IP 10.10.1.1 > 10.10.1.2: ICMP echo request, id 2760, seq 4, length 64
21:57:31.854626 IP 10.10.1.1 > 10.10.1.2: ICMP echo reply, id 2760, seq 4, length 64
21:57:32.855269 IP 10.10.1.1 > 10.10.1.2: ICMP echo request, id 2760, seq 5, length 64
21:57:32.855766 IP 10.10.1.1 > 10.10.1.2: ICMP echo reply, id 2760, seq 5, length 64
```

Figure 24: Packets observed from Switch4 or Switch6

```
root@switch4:~# tcdump -i eth1 -nq icmp
21:55:23.29120 IP 10.10.1.1 > 10.10.1.2: ICMP echo request, id 2752, seq 14, length 64
```

```
root@switch4:~# tcdump -i eth1 -nq icmp
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
6 packets captured
6 packets received by filter
0 packets dropped by kernel
```