Network Attack Experimentation using OpenFlow-enabled GENI Testbed

Mani Chaitanya Gorla, Venu Madhav Kamaraju, and Egemen K. Çetinkaya

Electrical and Computer Engineering Department, Missouri University of Science and Technology – http://web.mst.edu/~mstconets

I. INTRODUCTION and MOTIVATION

- **Introduction**

Computer networks require high bandwidth and dynamic management to support communication among distributed applications. While the Internet is widespread to provide infrastructure to users, it is prone to many malicious activities.

- **Types of attacks**

These malicious activities include FTP bounce attack, port scanning attack, ping flooding attack, smurf attack, IP fragmentation attack, IP sequence prediction attack, DNS cache poisoning, SNMP attack, teardrop attack, IP spoofing, fraggle attack, ping-of-death attack, timestamp attack, blind connection-reset attack, blind throughput-reduction, blind performance-degrading attack.

- In our study, based on the working principles of ICMP and TCP protocols, we perform two types of attacks on OpenFlow-enabled networks. They are:
  1. Ping flooding attack, which is done at the network layer
  2. TCP SYN attack, which is done at the transport layer
- We have chosen to perform these attacks because they are relatively easy to perform but effective in terms of resource consumption.

- **Importance of need for security**

- There are many malicious activities that interrupt the normal operations of networks.
- There have been incidents such as take down of banking websites, e-commerce websites, and trade market websites. This resulted a significant financial loss for both users and service providers. This increases our motivation for providing network security to the users and providers.
- A network has to be secure in order to have protection against internal and external network attacks, to ensure privacy for all communications, to have control over access to information by accurately identifying users, to provide authentication, and to be available to users.
- Today, it is non-trivial to conduct experiments on the newly developed protocols and their security threats. This requires huge network setup and a lot of financial support. This has led to the development of international testbeds, including the Global Environment for Network Innovations (GENI) funded by the National Science Foundation (NSF). We are using this test-bed to conduct our experiments.

II. RESEARCH GOALS

- Investigation of ping flood attack and TCP SYN attack against Software Defined Networks (SDN) with OpenFlow protocol
- Develop robust defense mechanisms to provide security measures at the control plane
- Develop schemes to detect the attacks at the control plane
- Experiment with network security and possible attacks against SDN
- Conduct experiments on a large scale testbed such as GENI

III. EXPERIMENTAL SETUP

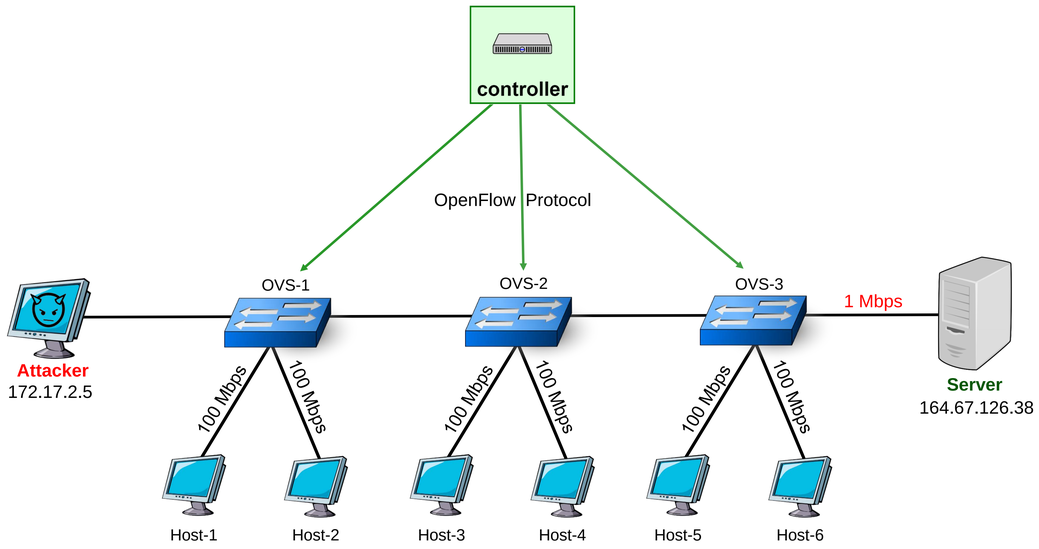

Figure 1. Experimental setup for ping flooding attack & TCP SYN attack

- **Topology description**

- We use three OpenFlow Virtual Switches (OVS), a POX controller, six hosts, an attacker, and a server. The attacker and hosts are loaded with Ubuntu 12 as operating system.
- The link capacity between these hosts is 100 Mbps, whereas the link capacity between server and OVS-3 is 1 Mbps.
- The GENI resources are reserved using Jacks (beta) and Flack tools on UtahDDC Instageni racks.

- **Tools used**

- **Wireshark:** an open-source packet analyser used to capture live data from a network and display the captured data with a GUI. It analyses different network layer protocols (ref: `https://www.wireshark.org`)
- **hping3:** a traffic generating tool to generate huge number of custom TCP/IP packets and is scriptable (ref: `http://www.hping.org/hping3.html`)
- **IPTraf:** a network monitoring tool in Linux based operating systems. It provides statistics of the traffic flowing through the interfaces (ref: `http://iptraf.seul.org`)
- **Custom scripts:** We developed Python scripts to perform TCP SYN attack on the server.

- **Attack experimentation**

- **Ping flooding attack:** Attacker sends `ICMP echo request` to the broadcast address of the network. Attacker spoofs his IP address as the victim's IP address by changing its `iptables` such that `ICMP echo reply` messages will be sent to server. As the request is sent to broadcast address, the request goes to all the nodes in the network. The nodes receive the echo request with the victim's IP address and they send an `ICMP echo reply` message to the victim. Attacker sends these echo requests at a fast rate and, correspondingly, nodes reply to the requests such that the link towards the victim is overwhelmed, thereby consuming the resources of the victim.
- We reserve resources, as shown in Figure 1. We place five compromised nodes (hosts 2-6). Next, we spoof the IP address of the attacker by setting it to that of the victim. The attacker uses the broadcast mechanism built into IPv4. The attacker sends the `ICMP echo request` message to the network's broadcast address. Then those hosts that receive this request send replies to the victim. As the requests are sent at a high rate, hosts also send replies at the same rate and the link towards the victim is congested.
- **TCP SYN attack:** TCP works on three-way handshake protocol to establish a connection. The attacker sends TCP `SYN` messages to the victim. The victim responds by sending the `SYN-ACK` to the attacker. Now the attacker will not send the `ACK` message to the victim. As a result the victim keeps the TCP connection open till timeout. Then the victim will not be able to handle any more TCP connections.
- We launch this by running a custom Python script on the attacker terminal. The attacker sends a huge number of TCP SYN messages to the server. The server responds to these requests by sending `SYN-ACK` messages. Next, to establish a connection, the attacker has to reply with `ACK` messages, which the attacker does not send. The server waits for the `ACK` messages from the attacker and keeps the TCP connection open until timeout. As a result, the server would cross the number of TCP connections it can handle, and it will not be able to make any further TCP connections.

IV. RESULTS and ANALYSIS

We have conducted the experiment using the topology shown in Figure 1. We obtained the results by running Wireshark on server and host-1. Figure 2a represents the traffic flow in the link between server and OVS-3. The link was not under attack up to 21 s and host-1 could get the ping reply from the server as shown in Figure 2b. After 21 s, the server is under attack and correspondingly host-1 could not get any reply from the server until 44th s. After stopping the attack, we are able to observe that the host-1 is able to receive replies from the server.

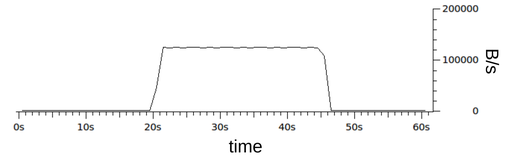

Figure 2a. Network traffic between OVS-3 and server

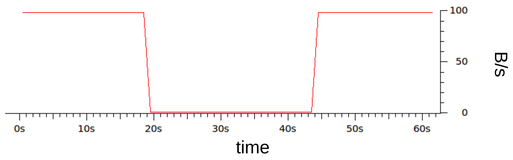

Figure 2b. ICMP replies from server to client (host-1)

Figure 3a shows a TCP three-way handshake between client and server under no attack. Figure 3b shows client-server interaction under attack. Client sends TCP SYN messages to server. Server sends ACK to the client but client do not send ACK to server and server keeps the TCP connection open till timeout.

```
|Time     | 172.17.2.5
|         |                  | 164.67.126.38
|3.628    |         SYN      |
|         |(40461)  ---------------->  (80)
|3.628    |         ACK      |
|         |(40461)  ---------------->  (80)
|3.628    |         PSH, ACK - Len: 131
|         |(40461)  ---------------->  (80)
|3.629    |         ACK      |
|         |(40461)  ---------------->  (80)
|3.631    |         FIN, ACK  |
|         |(40461)  ---------------->  (80)
|3.632    |         ACK      |
|         |(40461)  ---------------->  (80)
|5.668    |         SYN      |
|         |(40462)  ---------------->  (80)
|5.668    |         ACK      |
```

Figure 3a. Client-server interaction with no attack

```
|Time     | 172.17.2.5
|         |                  | 164.67.126.38
|0.000    |         SYN      |
|         |(55412)  ---------------->  (80)
|0.000    |         SYN, ACK  |
|         |(55412)  <----------------  (80)
|0.003    |         SYN      |
|         |(21379)  ---------------->  (80)
|0.003    |         SYN, ACK  |
|         |(21379)  <----------------  (80)
|0.006    |         SYN      |
|         |(21324)  ---------------->  (80)
|0.006    |         SYN, ACK  |
```

Figure 3b. Client-server interaction with attack

V. CONCLUSIONS and FUTURE WORK

- We performed a Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks against OpenFlow-enabled network and analyzed the effect of attack on the performance of the network.
- We will develop defense and detection mechanisms for the above mentioned attacks and apply that mechanism at the control plane of the network.