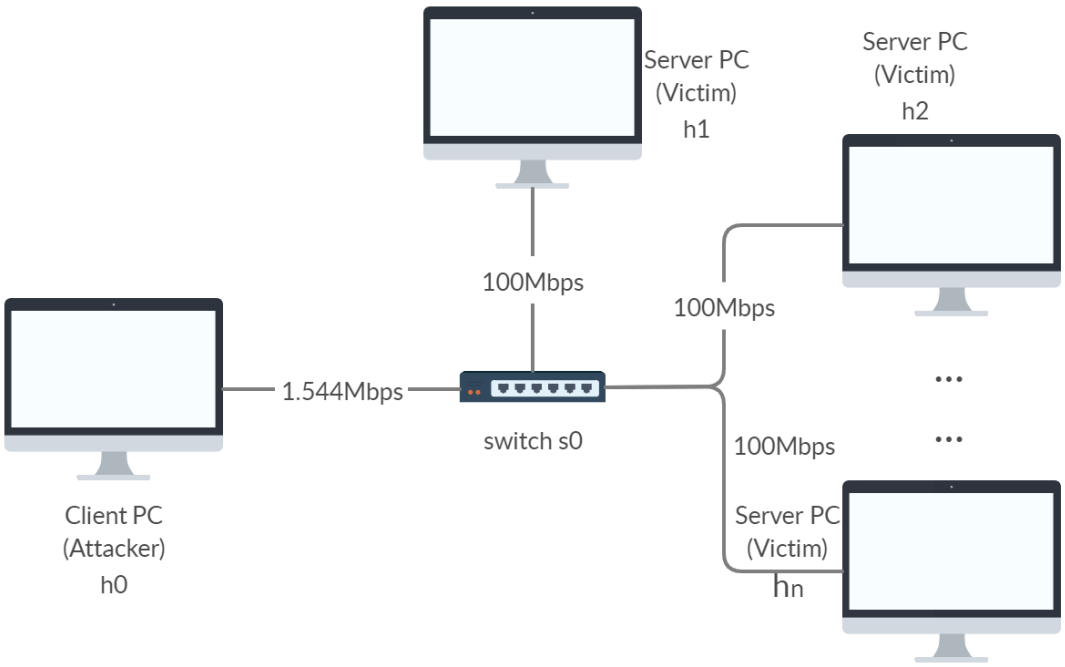
TCP Optimistic ACK Attack

Sifat Ishmam Parisa

Student ID – 1505016

Optimistic TCP ACK Attack

The underlying assumption of transmission control protocol - tcp, is the receiver end is trustworthy. As a result of the congestion control mechanism of tcp, the transmission rate is increased when there is no packet loss i.e positive ack. Henceforth, a misbehaving receiver who does not take data integrity into account could manipulate the sender into sending more packets even when data loss occurs by sending incorrect positive acknowledgment also known as optimistic acknowledgment or opt-ack. Opt ack attack can saturate the path from sender to receiver and potentially cause denial of service. The number of victims can be arbitrary.



Here the attacker PC is connected to n servers via a switch in a star topology.

Steps of the attack:

I have implemented tcp opt-ack attack using mininet. Mininet creates a **realistic virtual network**, running **real kernel, switch and application code**, on a single machine (VM, cloud or native). I have used mininet VM and the code scripts are in python. Increase of network traffic is shown in a plot by gnuplot.

Step 1:
Create Topology

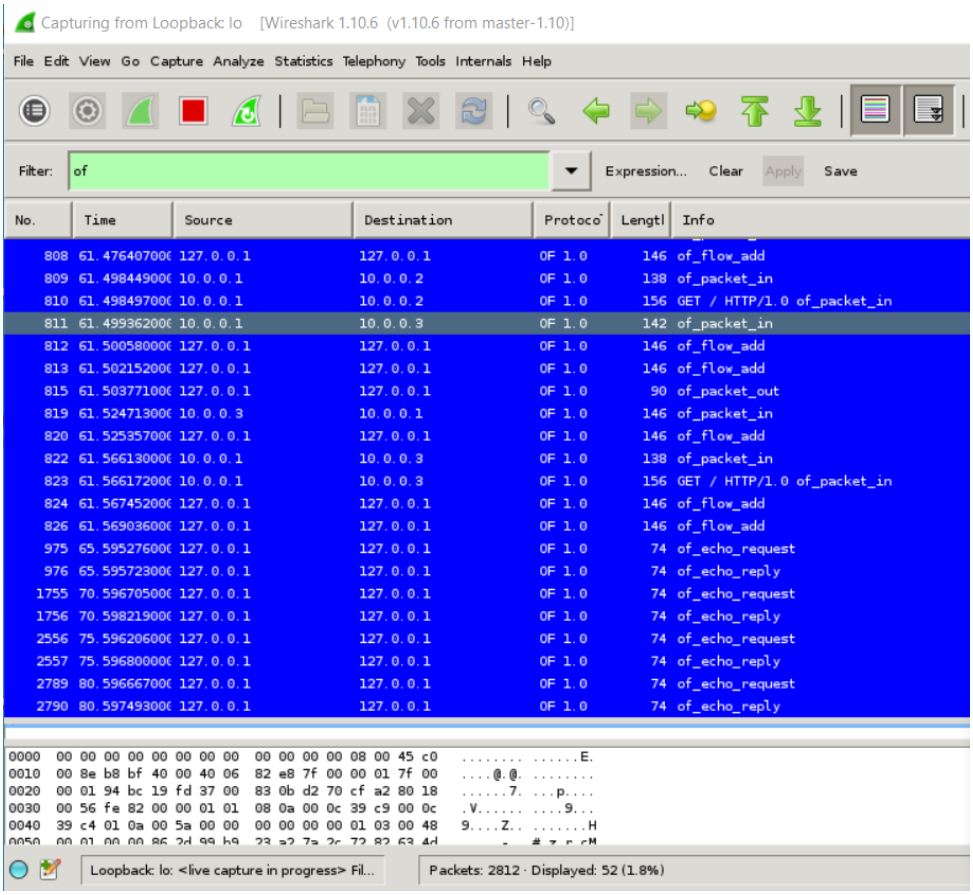
The client h0 is connected to switch s0 with bandwidth 1.544Mbps . Then for n=1,2,4,8,16,32,64 servers h1, h2…, hn are connected to switch s0 with bandwidth 100Mbps.

Step 2:
Start net

The servers sends packets to the client until termination. The client receives the data from all servers and sends ack packets but not cause overrun (send ack for a packet that the server has not yet sent).
Mininet CLI:

```
Client Address 10.0.0.1
Server Addresses 10.0.0.2 8080
Experiment with 1 server finished
Client Address 10.0.0.1
Server Addresses 10.0.0.2 8080 10.0.0.3 8080
Experiment with 2 servers finished
Client Address 10.0.0.1
Server Addresses 10.0.0.2 8080 10.0.0.3 8080 10.0.0.4 8080 10.0.0.5 8080
Experiment with 4 servers finished
Client Address 10.0.0.1
Server Addresses 10.0.0.2 8080 10.0.0.3 8080 10.0.0.4 8080 10.0.0.5 8080 10.0.0.
6 8080 10.0.0.7 8080 10.0.0.8 8080 10.0.0.9 8080
Experiment with 8 servers finished
Client Address 10.0.0.1
Server Addresses 10.0.0.2 8080 10.0.0.3 8080 10.0.0.4 8080 10.0.0.5 8080 10.0.0.
6 8080 10.0.0.7 8080 10.0.0.8 8080 10.0.0.9 8080 10.0.0.10 8080 10.0.0.11 8080 1
0.0.0.12 8080 10.0.0.13 8080 10.0.0.14 8080 10.0.0.15 8080 10.0.0.16 8080 10.0.0
.17 8080
Experiment with 16 servers finished
Client Address 10.0.0.1
Server Addresses 10.0.0.2 8080 10.0.0.3 8080 10.0.0.4 8080 10.0.0.5 8080 10.0.0.
6 8080 10.0.0.7 8080 10.0.0.8 8080 10.0.0.9 8080 10.0.0.10 8080 10.0.0.11 8080 1
0.0.0.12 8080 10.0.0.13 8080 10.0.0.14 8080 10.0.0.15 8080 10.0.0.16 8080 10.0.0
.17 8080 10.0.0.18 8080 10.0.0.19 8080 10.0.0.20 8080 10.0.0.21 8080 10.0.0.22 8
080 10.0.0.23 8080 10.0.0.24 8080 10.0.0.25 8080 10.0.0.26 8080 10.0.0.27 8080 1
0.0.0.28 8080 10.0.0.29 8080 10.0.0.30 8080 10.0.0.31 8080 10.0.0.32 8080 10.0.0
.33 8080
```

Wireshark was started with loopback and OF filter. Wireshark output:



Step 3:

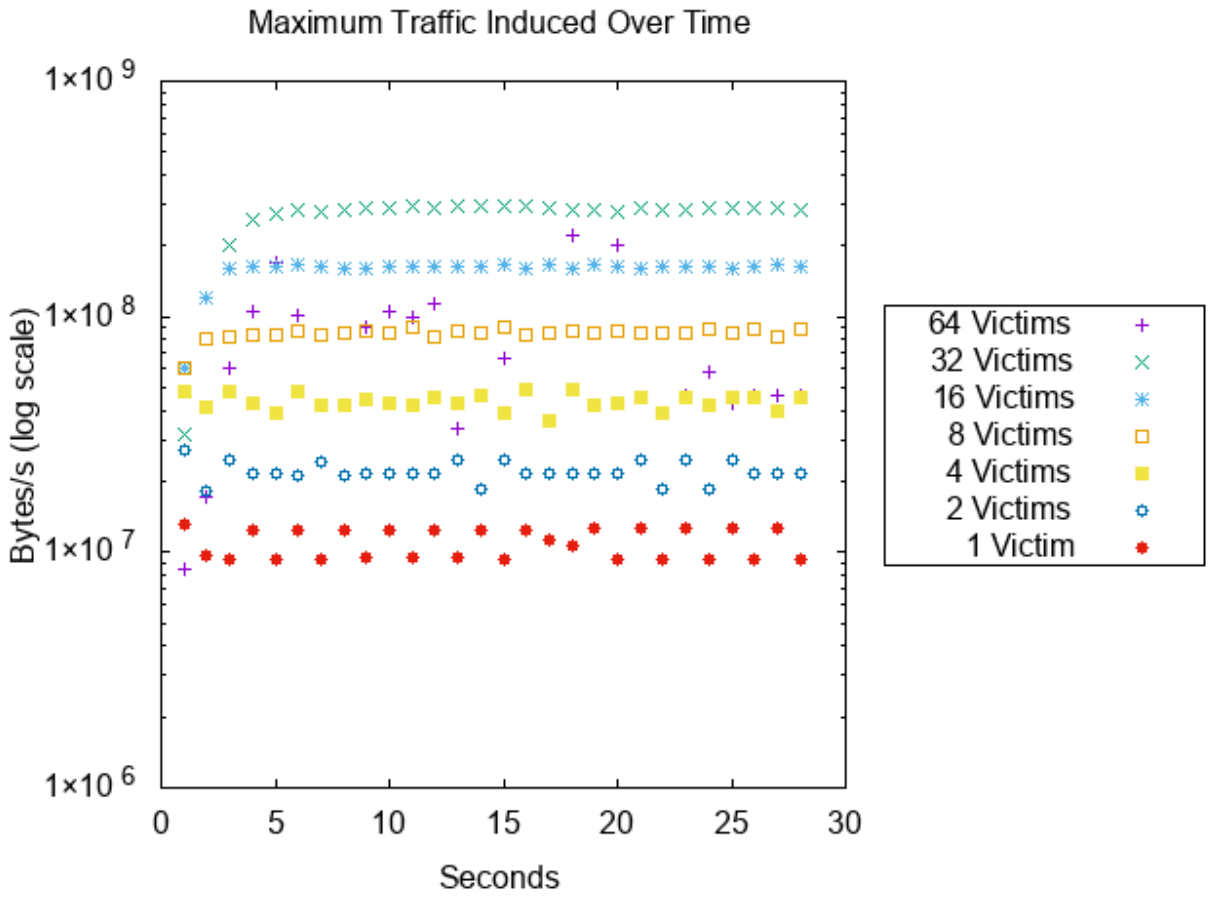
<u>Log output:</u>

Time and size(bytes) of sent data are logged in csv files

16.csv

```
1, 60280000
2, 119360000
3, 158940000
4, 163960000
5, 164560000
6, 165500000
7, 162160000
8, 160600000
9, 161120000
10, 163640000
11, 162100000
12, 164660000
13, 162080000
14, 164580000
15, 165920000
16, 160600000
17, 165080000
18, 160760000
19, 166920000
20, 164580000
21, 160120000
22, 164540000
23, 162660000
24, 162080000
25, 159100000
26, 162820000
27, 165000000
28, 163380000
```

Step 4:
Plot:



Success of the attack:

From the plot it is evident that opt-ack causes increase in traffic in the network. Packets received from the victim and retransmission detection are used to estimate congestion window and avoid overrun. Because of measurement issues there are fluctuations in the graph with less victims.

Countermeasures:

I did not implement any countermeasure. However, the following could be applied to defend against opt ack:

Selectively Send Packets:

The server can randomly choose not to send a packet and mark that packet. When a malicious client sends ack for that packet the server would therefore know the client is malicious.