Simulating TCP connection using scapy

Introduction

In this assignment, you will simulate a TCP client using scapy instead of the socket library. Scapy is a python library that can be used for creating network packets. Be sure to read the submission guidelines for the assignment and remember the rules applicable to all assignments. Also, go through the scapy tutorial before attempting this assignment.

Note: This assignment is manually evaluated at specific times and not automatically graded. However, you will get feedback if your submission is valid or not. Please refer to the assignment announcement for more details.

Question 1: Simulating a TCP client

You will create a TCP client using scapy that performs the following steps:

1. Initiate a connection to the TCP server using TCP 3 way connection establishment handshake.
2. Read 1 line of data from server.
3. Send the SHA256 hash of the server data(in hexadecimal format) terminated by a newline.
4. Complete the 4 way TCP connection termination handshake initiated by the server.

TCP 3 way connection establishment handshake

Recall that there are 3 steps in TCP to establish a connection:

1. Client sends a TCP SYN packet to the server.
2. Server sends a TCP SYN ACK packet as response to the client.
3. Client acknowledges the TCP SYN ACK packet with an ACK packet of it's own.

You will write a client which initiates the 3 way handshake with a server running on localhost, by creating the appropriate packets using scapy, with the correct values for all the fields.

Reading server data and sending SHA256 hash

The server data will be contained in a TCP packet as a Raw scapy packet. Thus, you have to use scapy to extract the string sent by the server. Similarly, you have to construct a Raw scapy packet from the SHA256 hash of server data and send it as the payload of a TCP packet. Use the hashlib library of Python to compute the SHA256 hash of the data. Remember to send the SHA256 hash to the server in hexadecimal format.

TCP 4 way connection termination handshake

Recall that there are 4 steps in TCP to terminate a connection:

1. Sender sends a TCP FIN packet.
2. Receiver acknowledges the TCP FIN packet with an ACK packet.

3. Receiver sends a TCP FIN packet.
4. Sender acknowledges the TCP FIN packet from receiver with an ACK packet.

We will ignore the fifth TIME_WAIT step where both sender and receiver wait for an interval before closing the connection.

For this assignment, the server will initiate the connection termination on receiving the SHA256 hash of server data from the client. Thus, your client should capture the FIN packet from server(step 1), respond to it(step 2) and then send it's own FIN packet to the server(steps 3 and 4).

How we will test your program

We will pass 3 arguments to your solution client program:

1. Source IP to use in the program.
2. Destination IP i.e. IP address of the server.
3. TCP port of the server.

An example invocation is shown below:

```
$ python2 client.py 192.168.56.2 192.168.56.1 8080
```

In above invocation, the server program to which client must connect to is running at port 8080 on IP address 192.168.56.1 and the client must use 192.168.56.2 as the source IP address. You can assume that the server will be running before the client is invoked.

Important: Note that this assignment is not automatically graded. It is instead graded manually at specific times. However, you will get feedback if you submission is valid or not. Please check the assignment announcement for more details.

How you can test your program

You can use *nc* command to simulate a server against which you can test your client. You capture packets on the local interface(*lo*) to check the values of the packets sent and received. Alternatively, you can print packets received and sent from within the client program for debugging. Refer to the scapy tutorial for how to print packets out.

Things to remember

A few things to remember when writing a client in scapy.

1. Remember to set the interface name and L3socket in scapy configuration. Refer to the scapy tutorial for more on this.
2. Remember to disable RST packets sent from the kernel using iptables before testing your solution. Refer to the scapy tutorial for more on this.