# CSE 406: Computer Security
# TCP Session Hijacking Attack

Bishal Basak Papan
Student ID: 1505043

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
(BUET)
Dhaka 1000

July 30, 2019

## Contents

## 1 Introduction

### 1.1 About TCP Protocol

The Transmission Control Protocol (TCP) is a core protocol of the Internet protocol suite. It sits on top of the IP layer, and provides a reliable and ordered communication channel between applications running on networked computers. Most applications such as browsers, SSH, Telnet, and email use TCP for communication. TCP is in a layer called Transport layer, which provides host-to-host communication services for application.

### 1.2 How TCP Protocol Works

To achieve reliability and ordered communication, TCP requires both ends of a communication to maintain a connection. Although this connection is only logical, not physical, conceptually we can imagine this connection as two pipes between two communicating applications, one for each direction: data put into a pipe from one end will be delivered to the other end.

Once a connection is established, the operating system allocates two buffers for each end, one for sending data (send buffer), and other for receiving data (receive buffer). TCP is duplex, both ends can send and receive data.
When an application needs to send data out, it does not construct a packet directly; instead, it places data into the TCP send buffer. The TCP code inside the operating system decides when to send data out. To avoid sending packets with small data and therefore waste network bandwidth, TCP usually waits for a little bit, such as 200 milliseconds, or until the data are enough to put into one packet without causing IP fragmentation.
Each octet in the send buffer has a sequence number associated with it. Inside the TCP header, there is a field called sequence number, which indicates the sequence number of the first octet in the payload. When packets arrive at the receiver side, TCP uses these sequence numbers from the TCP header to place data in the right position inside the receive buffer.

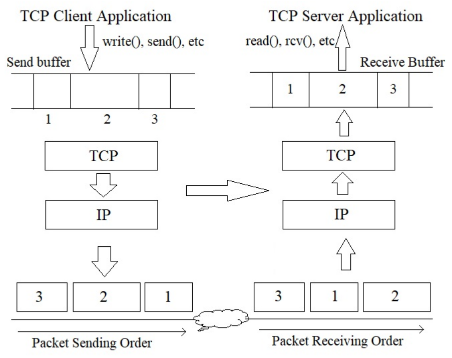
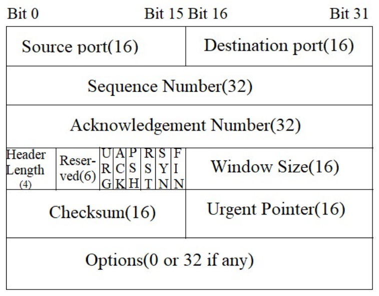
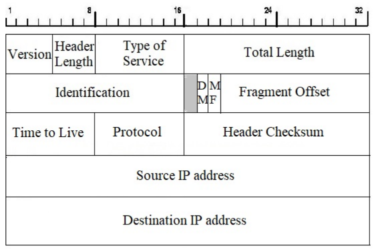


Figure: Data Transmission: Under the Hood

## 1.3 TCP and IP Headers



(a) Figure: TCP Header Format



(b) Figure: IP Header Format

# 2 TCP Session Hijacking Attack

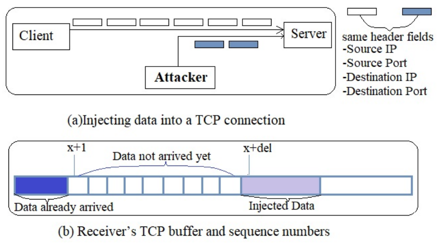


Figure: TCP Session Hijacking Attack

When a connection is established between two hosts, the connection is supposed to be used only by these two hosts. If an attacker can inject his/her own data into this connection, the connection can essentially be hijacked by the attacker, and its integrity will be compromised.

## 2.1 About TCP Session

Once a TCP client and server finish the three-way handshake protocol, a connection is established, and we call it a TCP session. From then on, both ends can send data to each other. Since a computer can have multiple concurrent TCP sessions with other computers, when it receives a packet, it needs to know which TCP session the packet belongs to. TCP uses four elements to to uniquely identify a session:

- source IP address

- destination IP address
- source port number
- destination port number.

These four fields are called the signature of a TCP session. If the above four elements match with the signature of the session, the receiver cannot tell whether the packet comes from the real sender or an attacker, so it considers the packet as belonging to the session. However, for the packet to be accepted, one more critical condition needs to be satisfied. It is the TCP sequence number.

## 2.2 Proposed Tool

### 2.2.1 Features of Tool

We consider that the User system, the Server system and the Attacker system all are in the same LAN.

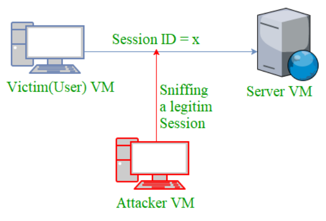
- A user (the victim) first establishes telnet connection from User to Server
- Attacker will sniff packets from client to server and get the details i.e. TCP headers of the packets
- Attacker will then construct packets by modifying TCP and IP headers. Attacker will set Server's IP address as Destination IP address, its own IP address as Source IP address, Server's Port as Destination port, its own port as source port and sequence number within the receiving window.
- In this way, the attacker will hijack this connection, and run an arbitrary command on Server, using the victim's privilege.

### 2.2.2 Necessary Arrangements for Implementation

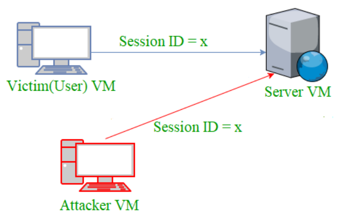
- To see a TCP session hijacking attack in action, we will launch it in our Virtual Machine(VM) environment. We set up 3 VMS: User, Server and Attacker.
- We shall implement the basic code-base in Python language, taking help from scapy package.
- We will use Wireshark to sniff packets
- In the spoofed packet, we need to set the ACK bit to 1, while putting the correct acknowledgment number in the TCP header.
- Sometimes, it may be difficult to get the exact sequence number in an attack, especially if the victim is still typing in the client terminal. In this case, we can make an estimate; for example, if we see an sequence number N for now, we can use N + 100 in the attack. As long as the data is within the server's receive window, our spoofed data will be placed in the receiver's buffer.

## 3 Diagrams

### 3.1 Attack Diagram



(a) Figure: Attacker Attempting to hijack session



(b) Figure: System Under Attack

### 3.2 Timing Diagram

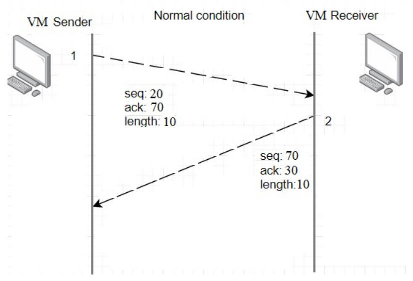


Figure: Initial State

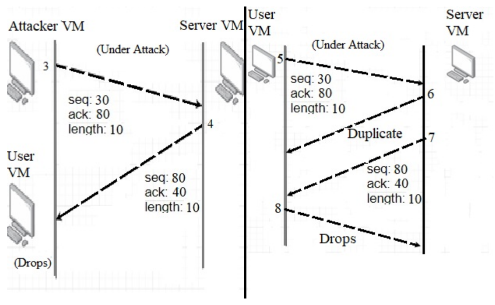


Figure: Under Attack State

## 4 Justification

- Our designed system will not work if all the three Virtual Machines are not in the same LAN.
- As long as the predicted sequence number doesn't fall outside of the bound of the buffer at the Server end, the packet will be received by the Server and acknowledged for.
- When a TCP packet is spoofed, the sequence number field of the TCP header needs to be set appropriately. The server VM has already received some data up to the sequence number x, so the next sequence number is x + datagram_length. If the spoofed packet does not set x + datagram_length as its sequence number, and instead uses x + $\delta$, where $\delta >$ datagram_length this becomes an out-of-order packet. The data in this packet will be stored in the receiver's buffer (as long as the buffer has enough space), but not at the beginning of the free space; it will be stored at position $x + \delta$, leaving $\delta$ spaces in the buffer. The spoofed data will stay in the buffer, not delivered to the application, until the missing space is filled by future TCP packets. If $\delta$ is too large, it may fall out of the buffer boundary, and the spoofed packet will be discarded.
- If the correct acknowledgment number is assumed, the command sent in the payload will be sent to the application at the receiver end and executed right away.