2.9 TCP session hijacking

TCP session hijacking

TCP

Before seeing the **TCP session hijacking attack**, we need to understand a bit more about the TCP Protocol of the Transport Layer of the OSI model.

The **Transmission Control Protocol** (TCP) is one of the main protocols that we can find in the OSI model.

The TCP packet format consists of the following fields:

- **Source Port and Destination Port fields** (16 bits each) identify the endpoints of the connection.
- **Sequence Number field** (32 bits) specifies the number assigned to the first byte of data in the current message. Under certain circumstances, it can also be used to identify an initial sequence number to be used in the upcoming transmission.
- **Acknowledgement Number field** (32 bits) contains the value of the next sequence number that the sender of the segment is expecting to receive, if the ACK control bit is set. Note that the sequence number refers to the stream flowing in the same direction as the segment, while the acknowledgement number refers to the stream flowing in the opposite direction from the segment.
- **Data Offset field** (variable length), also known as Header Length, tells how many 32-bit words are contained in the TCP header. This information is needed because the Options field has a variable length, so the header length is variable too.
- **Reserved field** (6 bits) must be zero. This is for future use.
- **Flags field** (6 bits) contains the various flags:
- **URG** - Indicates that some urgent data have been placed.
- **ACK** - Indicates that acknowledgement number is valid.
- **PSH** - Indicates that data should be passed to the application as soon as possible.
- **RST** - Resets the connection.
- **SYN** - Synchronises sequence numbers to initiate a connection.
- **FIN** - Means that the sender of the flag has finished sending data.

- **Window field** (16 bits) specifies the size of the sender's receive window (that is the buffer space available for incoming data).
- **Checksum field** (16 bits) indicates whether the header was damaged in transit.
- **Urgent pointer field** (16 bits) points to the first urgent data byte in the packet.
- **Options field** (variable length) specifies various TCP options.
- **Data field** (variable length) contains upper-layer information.

TCP connection hijacking

To establish a connection, we know that the Three-Way Handshaking process negotiates the opening of a new connection between two parties.

A scenario that has been called **TCP connection hijacking** sees the attacker trying to hijack the connection between a client and a server.

If the timing of the attacker is correct, it would be able to alter the TCP stream without the receiver realising the alteration. However, in order to do so, the attacker needs to know the sequence number. The prediction of the sequence number is not trivial, if the attacker is **on-path** (within the network). In fact, in this case, the steps are:

1. Sniff the traffic collecting packets.
2. Predict the sequence number (checking the packets collected).
3. Inject the data.

Here we see the diagram from Three-Way Handshake; however, this time an attacker has hijacked the connection. The second packet from the server (the last arrow on the diagram) is ignored as the client has already processed that particular sequence number.

Let’s assume that the attacker is able to hijack a TCP stream. How can the attacker use it?

To spoof the client: the attacker can decide to hijack the connection right after the initial authentication phase and play the role of the client. The attacker can send packets that will be considered from an 'authenticated' user.

To spoof the server: the attacker could play the role of the server, as we saw before, injecting false data inside the stream, potentially redirecting the client to specific pages the attacker created.

However, what could an attacker do if it is **off-path**, that is outside of the network? The attacker could use IP spoofing against the client sending a packet to the server, pretending to be the client. For this to work, the attacker's packets have to be in the right part of the stream, otherwise the server will ignore the packets.

The attacker can generate the first message of the sequence, pretending to be the client. However, when the server responds, the message will be sent to the real client. There are two observations we need to do.

Firstly, the client may send an RST (Reset) message as the client knows that it never initiated a connection, and the server will terminate the connection. Alternatively, since the attacker never receives the message, it won't be able to get to see Y (the sequence number generated by the server that it wants to use for this connection). So, in order to keep trying to open the connection and send the subsequential message, the attacker needs to guess Y+1.

Why doesn't the attacker see Y, the sequence number sent by the server to the client?

How do the sequence numbers get generated according to the RFC 793 (TCP Standard)?

View answer

The TCP standard says to use the clock.

When new connections are created, an **initial sequence number** (ISN) generator is employed which selects a new 32 bit ISN. The generator is bound to a (possibly fictitious) 32-bit clock whose low order bit is incremented roughly every 4 microseconds. Thus, the ISN cycles approximately every 4.55 hours. Since we assume that segments will stay in the network no more than the Maximum Segment Lifetime (MSL) and that the MSL is less than 4.55 hours we can reasonably assume that ISNs will be unique. That is the sequence numbers should not repeat for at least 4.55 hours. This is to ensure undelivered packets from previous connections that arrive late do not overlap with current connections.

Is it then possible for the attacker to guess the initial sequence number? The answer is yes. Since the protocol is open, all the specifications are public and it is possible to calculate the sequence number as it is just a mathematical operation.

Even if the protocol is public, some operative systems can decide to implement it differently, as BSD did. So, in this case, what can the attacker do? For instance, what if the attacker created a legitimate connection with the server before spoofing the client? The attacker will see what is the initial sequence number the server wants to use with it and it could easily guess what the next sequence number for another connection would be based on the implementation.

Note on the BSD acronym

Berkeley Software Distribution (BSD) was an operating system based on Research Unix, developed and distributed by the Computer Systems Research Group (CSRG) at the University of California, Berkeley.

Today, 'BSD' often refers to its descendants, such as FreeBSD, OpenBSD, NetBSD, or DragonFly BSD, and systems based on those descendants.