Nick Weaver
Fall 2018

CS 161
Computer Security

Discussion 8

Week of October 15, 2018

Question 1 ***DHCP*** **(5 min)**

Nick gets home after a tiring day of lecturing CS 161. He opens up his laptop and goes on Twitter. From a networking and web perspective, what are the steps involved in loading the Twitter homepage?

Nick's computer needs to connect to the wifi. What messages are exchanged in the 4 part handshake in order to achieve this?

Nick's computer sends: _____.

This message is *broadcasted / unicasted*. Choose one and explain:

A DHCP server replies with a DHCP Offer. What does this message contain? What can a malicious attacker do at this step? Keep in mind that an attacker on the same subnet can hear the discovery message.

Nick's computer sends: _____.

This message is *broadcasted / unicasted*. Choose one and explain:

The server then responds with: _____.

Solution: DHCP discover, broadcast. New host does not have an IP address and does not know what destination address to use.

The DHCP Offer message includes IP address, DNS server, gateway router, and lease time. Attacker can race the actual server; if they win, they can replace the DNS server and/or gateway router. Substitute a fake DNS server: Redirect any of a hosts lookups to a machine of attackers choice. Substitute a fake gateway router: Intercept all of a hosts off-subnet traffic (even if not preceded by a DNS lookup). Relay contents back and forth between host and remote server and modify however attacker chooses. An invisible Man In The Middle (MITM): Victim host has no way of knowing its happening (Cant necessarily alarm on peculiarity of receiving multiple DHCP replies, since that can happen benignly)

DHCP request, broadcast. Multiple DHCP servers can send out DHCP offers, but the client only accepts one. The broadcast tells the other DHCP servers which offer

the client has accepted, and the rest of them can withdraw their offers that were not accepted and return the offered addresses to the pool of available addresses.

DHCP ACK.

Question 2 ***Back to L4 Basics*** **(10 min)**

The transmission control protocol (TCP) and user datagram protocol (UDP) are two of the primary protocols of the Internet protocol suite.

(a) How do TCP and UDP relate to IP (Internet protocol)? Which of these protocols are encapsulated within (or layered atop) one another? Could all three be used simultaneously?

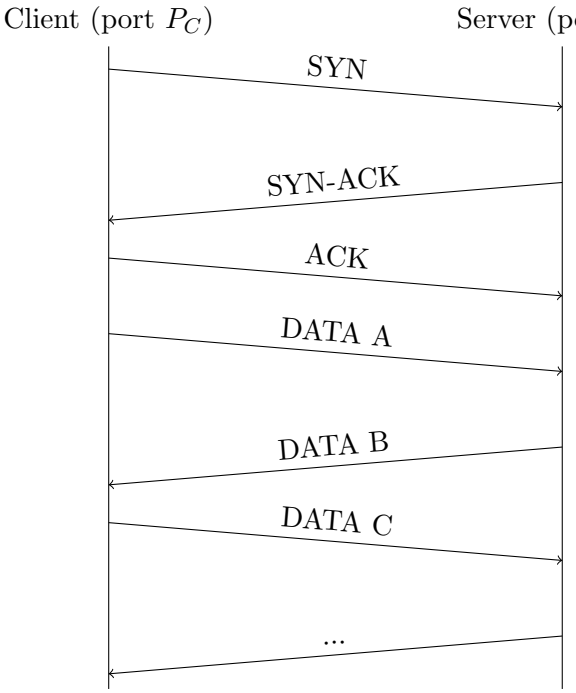
> **Solution:** TCP and UDP both exist within the transport layer, which is one layer above IP (network layer). Either can be encapsulated in IP, referred to as TCP/IP and UDP/IP. TCP and UDP are alternatives; neither would normally be encapsulated within the other.

(b) What are the differences between TCP and UDP? Which is considered "best effort"? What does that mean?

> **Solution:** TCP provides a *connection-oriented*, *reliable*, *bytestream* service. It includes sophisticated rate-control enabling it to achieve high performance but also respond to changes in network capacity. UDP provides a *datagram-oriented*, *unreliable* service. (Datagrams are essentially individual packets.) The main benefit of UDP is that it is lightweight.
>
> "Best effort" refers to a delivery service that simply makes a single attempt to deliver a packet, but with no guarantees. IP provides such a service, and because UDP simply encapsulates its datagrams directly into IP packets with very little additional delivery properties, it, too, provides "best effort" service.

(c) Which is easier to spoof, and why?

> **Solution:** Spoofing a UDP packet requires determining the correct port numbers to use, but no more. Spoofing a TCP packet requires both correct port numbers and also correct sequence numbers, making it significantly more difficult.



1. Client sends initial SYN with sequence number A (usually random).
2. Server sends SYN-ACK with sequence number B (also usually random) and ACK $A+1$.

3. Client sends ACK with sequence number $A+1$ and ACK $B+1$.
4. Client sends DATA A of length L_A with sequence number $A+1$ and ACK $B+1$.
5. Server sends DATA B of length L_B with sequence number $B+1$ and ACK $A+1+L_A$.

6. Client sends DATA C of length L_C with sequence number $A+1+L_A$ and ACK $B+1+L_B$.
7. Data exchange continues until both sides are done sending data.

Figure 1: TCP handshake and initial data transfer

Question 3 ***Attack On TCP*** **(35 min)**

(a) Assume that the next transmission in this connection will be DATA D from the server to the client. What will this packet look like?

Sequence number:	$B+1+L_B$	ACK:	$A+1+L_A+L_C$
Source port:	P_S	Destination port:	P_C
Length:	L_D	Flags:	None

(b) You should be familiar with the concept and capabilities of a *man-in-the-middle* as an attacker who **CAN observe** and **CAN intercept** traffic. There are two other types of relevant attackers in this scenario:

1. *On-path* attacker: **CAN observe** traffic but **CANNOT intercept** it.
2. *Off-path* attacker: **CANNOT observe** traffic and **CANNOT intercept** it.

Carol is an *on-path* attacker. Can Carol do anything malicious to the connection? If so, what can she do?

Solution: Yes, Carol can leverage the information she learns from your traffic to hijack the session.

In part (a), we identified the values of all of the fields of concern expected in the next data transmission in the connection. Say Carol wants to spoof a

packet from the server to the client; Carol can create a packet with the source IP as the server's IP, the destination IP as the client's IP, and the payload as whatever she wants. To spoof traffic in the other direction, she swaps the sequence number/ACK, source port/destination port, and source IP/destination IP. The recipient of this data cannot distinguish it from legitimate traffic, so she has effectively hijacked the session from her victim, allowing her to inject arbitrary data.

(c) David is an *off-path* attacker. Can David do anything malicious to the connection? If so, what can he do?

Solution: No, there isn't much he can do.

In part (b), we demonstrated that we are effectively defenseless against an attacker that knows the *sequence numbers* and the *port numbers* of the connection. An off-path attacker, however, does not have the power to observe the traffic and find these parameters.

Even without prior knowledge of these parameters, though, an *off-path* attacker may attempt to guess them. In a typical TCP client-server connection, the client's port is an *ephemeral* port, with a maximum potential range of $[0, 2^{16} - 1]$ (this varies, so we make an overestimation). The server's port is usually a *well-known* port for a specific service, such as port 80 for HTTP, which makes it much easier to guess. The sequence number and acknowledgement numbers are in the range $[0, 2^{32} - 1]$. Thus, an attacker has a rough $\frac{1}{2^{80}}$ chance of successfully brute forcing the correct parameters.

If, for some reason, the initial sequence numbers are not properly randomized, David may be able to make educated guesses on the sequence numbers and significantly decrease the range of possibilities. However, assuming that they are properly randomized, this attack is theoretically possible but largely improbable.

We call this attack *blind hijacking*, as David has no concrete information when attempting to hijack the session.

(d) The client starts getting responses from the server that don't make any sense. Inferring that David is attempting to hijack the connection, the client then immediately sends the server a **RST** packet, which terminates the ongoing connection. Is the client now safe?

Solution: No, the client is not completely safe.

In part (c), we found that the space of 2^{80} possibilities was too large for David to effectively guess. Now, we consider the possibility that David attempts to create a new connection under his control instead of hijacking an existing connection. This operates under the assumption that the server *trusts* the client's IP address as an identifier of the client. If the server requires a secondary form of authentication, such as a session cookie, this attack is not plausible.

If David attempts to start a new connection, he can *choose* the source port (the *ephemeral* port) and the source sequence number to be whatever values he wants. The values of these parameters for any subsequent transmissions in the connection will then be predictable. The server's port remains a well-known port; the only remaining unknown is the acknowledgement number. Because

David is still an off-path attacker, he still has to guess this field, with an overall probability of $\frac{1}{2^{32}}$ of success, which we note is much higher than the *blind hijacking* approach.

Note that there's now a time constraint on David's attack: if the client receives a response from the server based on his spoofed *SYN*, it will send a **RST** and terminate the connection, putting David back at step 1.

We call this attack *blind spoofing*, as David has no concrete information when attempting to spoof a new session.