CS 161 Spring 2024 — Introduction to Computer Security — Exam Prep 10

Q1 *I (T)C(P) You (su20-final-q7)* **(26 points)**

EvanBot builds a new course feature that sends announcements to students over TCP. To receive announcements, a student initiates a TCP connection with the server. The server sends the announcements and terminates the connection.

Q1.1 (3 points) Assuming that no adversaries are present, which of the following does communication over a TCP connection guarantee? Select all that apply.

- ■ (A) That both the server and client can detect if a particular announcement needs to be resent
- ■ (B) That different announcements are delivered in the same order they were sent in
- ☐ (C) That announcements are delivered using the most efficient path through the internet
- ☐ (D) None of the above
- ☐ (E) ——
- ☐ (F) ——

Solution: TCP guarantees that messages will be retransmitted until they are successfully delivered, and that messages will be delivered in the correct order. TCP makes no guarantees about what path a packet takes through the Internet.

Q1.2 (3 points) When only an on-path adversary is present, which of the following does communication over a TCP connection guarantee? Select all that apply.

- ☐ (G) That both the server and client can detect if a particular announcement needs to be resent
- ☐ (H) That different announcements are delivered in the same order they were sent in
- ☐ (I) That announcements are delivered using the most efficient path through the internet
- ■ (J) None of the above
- ☐ (K) ——
- ☐ (L) ——

Solution: An on-path attacker has access to the TCP sequence numbers, so they can inject arbitrary messages. Since the attacker can interfere with all messages, TCP no longer has any guarantees about message delivery. TCP still makes no guarantees about what path a packet takes through the Internet.

Q1.3 (3 points) Suppose that EvanBot instead sends announcements over UDP. Assuming that no adversaries are present, which of the following might happen? Select all that apply.

■ (A) Students might not receive some announcements

■ (B) Students might receive the announcements more quickly

■ (C) The server might not detect some errors which it would have had it been using TCP

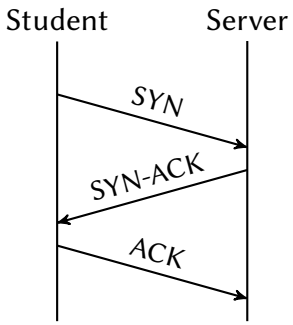
☐ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: UDP no longer guarantees delivery, so some announcements might not be delivered. However, UDP does not require a handshake at the beginning, so announcements can be delivered more quickly. UDP has no guarantees about what order announcements arrive in, so the server will no longer detect if packets arrive out of order.

EvanBot realizes that the server is sending messages to the student, but the student only responds with ACKs and never sends any messages after the initial handshake. They design a *Half TCP* protocol which provides TCP's properties for communications from the server to the student, but not for communications from the student to the server. This is accomplished using a modified version of the standard three step handshake pictured below.



Q1.4 (5 points) Some sequence numbers are no longer necessary in *Half TCP*. Which fields **do not** need to be transmitted? Select all that apply.

■ (G) The sequence number in the SYN packet

☐ (H) The sequence number in the SYN-ACK packet

■ (I) The ACK number in the SYN-ACK packet

■ (J) The sequence number in the ACK packet

☐ (K) The ACK number in the ACK packet

☐ (L) None of the above

Solution: The key insight here is that because the student isn't sending messages to the server, the student's sequence numbers are no longer necessary. The SYN and ACK packets are sent from the student to the server, so their sequence numbers are no longer necessary. The SYN-ACK packet is sent from the server to the student, so its ACK number is no longer necessary.

An earlier version of the solutions incorrectly marked H, K as the set of correct answers. When revising the exam, we changed the question to be "which fields **do not** need to be transmitted," which caused the set of correct answers to be inverted.

Q1.5 (3 points) Which of these are consequences of moving from TCP to *Half TCP* for this application? Select all that apply.

☐ (A) The student will no longer receive announcements in the correct order

■ (B) The server will not have to keep track of as much state

■ (C) The student will not have to keep track of as much state

☐ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: Announcements are sent from the server to the student. We are still using sequence numbers in this direction, so the announcements are still received in the correct order. Because the server and student each only need to keep track of one sequence number instead of two, they both do not need to keep track of as much state.

The 161 staff likes security and decides to use TLS over *Half TCP*. Assume that the staff server has a valid certificate for their public key.

For each different adversary below, select all attacks which become ***easier*** when running TLS over *Half TCP* compared to normal TCP.

Q1.6 (3 points) Off-path adversary

■ (G) RST Injection Attack

☐ (H) Interfere with a TLS handshake to learn the master key

☐ (I) Replay an encrypted command from a previous TLS connection

☐ (J) None of the above

☐ (K) ——

☐ (L) ——

Q1.7 (3 points) On-path adversary

☐ (A) RST Injection Attack

☐ (B) Interfere with a TLS handshake to learn the master key

☐ (C) Replay an encrypted command from a previous TLS connection

■ (D) None of the above

☐ (E) ——

☐ (F) ——

Q1.8 (3 points) Man-in-the-middle adversary

☐ (G) RST Injection Attack

☐ (H) Interfere with a TLS handshake to learn the master key

☐ (I) Replay an encrypted command from a previous TLS connection

■ (J) None of the above

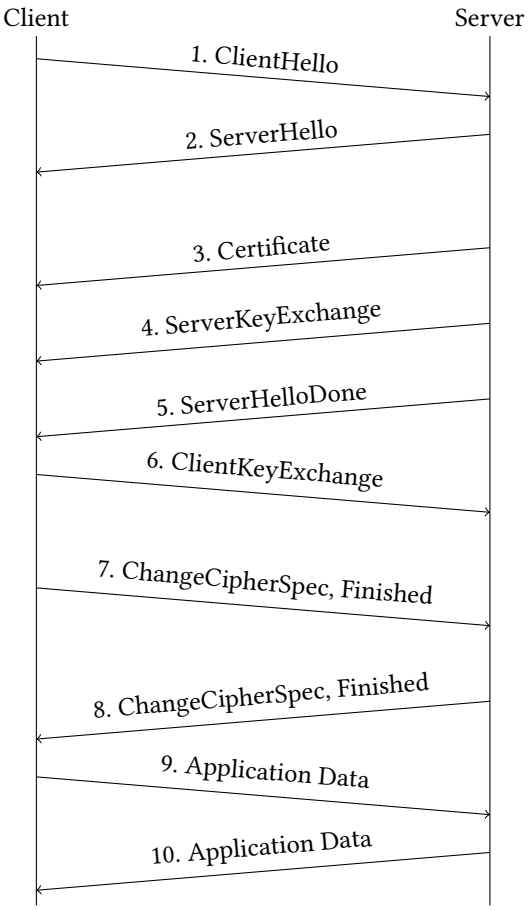
☐ (K) ——

☐ (L) ——

Solution: The key insight here is that attacks on the TLS protocol are not made any easier by using half-TCP, because the cryptographic messages sent between the student and the server are unchanged. The only attack that becomes easier is the RST injection attack for an off-path attacker, since the attacker doesn't need to guess sequence numbers when injecting a RST packet from the student to the server. On-path and MITM attackers can see all sequence numbers, so RST injection is not any easier for them.

Q2 *Mutuality (SP21 Final Q9)* **(18 points)**

Recall the TLS handshake:



1. Client sends 256-bit random number R_b and supported ciphers

2. Server sends 256-bit random number R_s and chosen cipher

3. Server sends certificate

4. DH: Server sends $\{g, p, g^a \bmod p\}_{K_{\text{server}}^{-1}}$

5. Server signals end of handshake

6. DH: Client sends $g^b \bmod p$
   RSA: Client sends $\{PS\}_{K_{\text{server}}}$
   Client and server derive cipher keys C_b, C_s and integrity keys I_b, I_s from R_b, R_s, PS

7. Client sends MAC(dialog, I_b)

8. Server sends MAC(dialog, I_s)

9. Client data takes the form $\{M_1, \text{MAC}(M_1, I_b)\}_{C_b}$

10. Server data takes the form $\{M_2, \text{MAC}(M_2, I_s)\}_{C_s}$

In TLS, we verify the identity of the server, but not the client. How would we modify TLS to also verify the identity of the client?

Clarification during exam: All parts of this question refer to a modified TLS scheme designed to verify the identity of the client.

Q2.1 (3 points) Which of these additional values should the client send to the server?

◯ (A) A certificate with the client's public key, signed by the client's private key

◯ (B) A certificate with the client's public key, signed by the server's private key

◯ (C) A certificate with the client's private key, signed by a certificate authority's private key

● (D) A certificate with the client's public key, signed by a certificate authority's private key

◯ (E) —

◯ (F) —

Solution: This is analogous to the server sending its certificate, which has the server's public key, signed by the certificate authority's private key.

Q2.2 (3 points) How should the client send the premaster secret in RSA TLS?

● (G) Encrypted with the server's public key, signed by the client's private key

◯ (H) Encrypted with the client's public key, signed by the server's private key

◯ (I) Encrypted with the server's public key, signed by a certificate authority's private key

◯ (J) Encrypted with the client's public key, signed by a certificate authority's private key

◯ (K) —

◯ (L) —

Solution: The client should encrypt the premaster secret with the server's public key so that the server can decrypt it (just like in regular TLS).

However, the client should additionally sign the premaster secret, so that the server can validate the signature and confirm that the server is talking to the correct client. The client should sign the premaster secret with their own private key. (The client doesn't know the certificate authority's private key, and the CA's private key is only used to sign certificates anyway.)

Q2.3 (3 points) EvanBot argues that the key exchange protocol in Diffie-Hellman TLS doesn't need to be changed to support client validation. Is EvanBot right?

◯ (A) Yes, because only the client knows the secret a, so the server can be sure it's talking to the legitimate client

◯ (B) Yes, because the server has already received and verified the client's certificate

● (C) No, the client must additionally sign their part of the Diffie-Hellman exchange with the client's private key

◯ (D) No, the client must additionally sign their part of the Diffie-Hellman exchange with the certificate authority's private key

◯ (E) ——

◯ (F) ——

Solution: Diffie-Hellman on its own doesn't provide any authenticity. We also need the client to sign their Diffie-Hellman message.

Q2.4 (2 points) TRUE or FALSE: The server can be sure that they're talking to the client (and not an attacker impersonating the client) immediately after the client and server exchange certificates.

◯ (G) True ● (H) False ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Solution: False. Remember that certificates are public, and attackers can present a certificate for anyone. The ClientHello and ServerHello messages only contain random nonces and an agreement on what algorithms to use, so they also do not give the client and server any guarantees about who they're talking to.

The client and the server need to wait at least until the signatures are exchanged to verify that they're talking to the correct person. If an attacker tampers with the handshake, the client and the server may even have to wait until the MACs are exchanged.

Q2.5 (3 points) At what step in the TLS handshake can both the client and server be sure that they have derived the same symmetric keys?

◯ (A) Immediately after the TCP handshake, before the TLS handshake starts

◯ (B) Immediately after the ClientHello and ServerHello are sent

◯ (C) Immediately after the client and server exchange certificates

◯ (D) Immediately after the client and server verify signatures

● (E) Immediately after the MACs are exchanged and verified

◯ (F) ——

Solution: The reasoning here is the same as in regular TLS. A MITM could tamper with messages, and the client and server will only detect this once they verify the MAC on the entire handshake.

Q2.6 (4 points) Which of these keys, if stolen individually, would allow the attacker to impersonate the client? Select all that apply.

■ (G) Private key of a certificate authority

■ (H) Private key of the client

☐ (I) Private key of the server

☐ (J) Public key of a certificate authority

☐ (K) None of the above

☐ (L) ——

Solution: If the attacker steals the private key of a trusted CA, they can sign a fake certificate claiming that the attacker's public key belongs to the client.

If the attacker steals the private key of the client, they can sign messages as the client.

Stealing the public key of the server doesn't help the attacker impersonate the client.

Q3 *Networking: TLS Times Two* **(14 points)**

A client and server form a secure connection with Diffie-Hellman TLS. The client uses Diffie-Hellman secret c_1, and the server uses secret s_1. After the first connection ends, Mallory, a MITM attacker, compromises s_1.

Next, the same client and server form a second connection with Diffie-Hellman TLS. For this connection, the client uses Diffie-Hellman secret c_2, and the server uses secret s_2.

Mallory wants to impersonate the server in the second connection (i.e. Mallory wants to be able to send her own messages to the client in the second connection).
Clarification during exam: Assume that Mallory recorded all communication from the first TLS connection.

Q3.1 (3 points) During the second handshake, the server sends $g^{s_2} \bmod p$ to the client, along with a signature on this value.

Mallory intercepts this message and replaces it, sending the replaced message to the client. What should the replaced message be?

Your answer can contain any values that Mallory knows.

Solution: Mallory sends $g^{s_1} \bmod p$ and a signature on that message.

The value $g^{s_1} \bmod p$ either comes from Mallory knowing s_1 and computing it herself, or from Mallory recording the first handshake.

The signature on $g^{s_1} \bmod p$ comes from Mallory recording the first handshake.

Q3.2 (3 points) What is the shared premaster secret that the client derives?

Solution: $g^{s_1 c_2} \bmod p$

The client is still using their own Diffie-Hellman secret c_2, but receives the old Diffie-Hellman secret s_1 from the server because Mallory replayed it.

Q3.3 (3 points) After executing this attack, what can Mallory do in the second TLS connection? Select all that apply.

- ■ Read messages sent by the client
- ☐ Send messages to the server
- ☐ Read messages sent by the server
- ☐ None of the above

Solution: The client is encrypting messages with a key derived from $g^{s_1 c_2} \bmod p$, which Mallory knows.

The server is encrypting messages with a key derived from $g^{s_2 c_2} \bmod p$, which Mallory doesn't know, since she only changed the key seen by the client. Therefore, Mallory cannot read messages sent by the server, or send messages to the server (since Mallory doesn't know what key to encrypt those messages with).

Q3.4 (5 points) Suppose the server acts as a certificate authority for EvanBot. (In other words, the server can use their secret key to sign EvanBot's public keys.)

The client wants to form a TLS connection with EvanBot. Can Mallory use this attack to cause EvanBot to derive a shared secret that Mallory knows?

○ Yes ● No

Briefly justify your answer.

Solution: Mallory cannot impersonate any child servers using a server certificate, as she does not know SK_{server} – despite being able to impersonate Google itself via replay attack.