Networking: TCP and TLS

Question 1 **(min)**

Q1.1 True or False: TLS has end-to-end security, so it is secure against an attacker who steals the private key of the server.

○ True ● False

Solution: False. An attacker who's stolen the private key of the server could impersonate the server to the victim.

Q1.2 True or False: By default, in a TLS connection, both the server and client are authenticated to each other.

○ True ● False

Solution: False. TLS only authenticates the server by default.

Q1.3 True or False: If the server's random number a in Diffie-Hellman TLS is the same in every handshake, Diffie-Hellman TLS no longer has forward secrecy. Assume the value a is stored on the server along with its secret key.

● True ○ False

Solution: True. An attacker who steals a will be able to reconstruct the PS and decrypt past recorded messages by computing $(g^b)^a \bmod p$.

Q1.4 True or False: Randomizing the client port helps defend TCP against on-path attackers.

○ True ● False

Solution: False. The on-path attacker can see the port values.

Q1.5 TRUE or FALSE: TLS provides end-to-end security, so it is secure even if the server has a buffer overflow vulnerability.

○ TRUE ● FALSE

Solution: False. If an attacker exploits the buffer overflow vulnerability to gain control of the server, TLS doesn't stop you from talking to the compromised server.

Q1.6 TRUE or FALSE: Suppose we modified TCP so that the sequence number increases by 2 for every byte sent, but the initial sequence numbers are still randomly chosen. This modified protocol has the same security guarantees as standard TCP.

● TRUE ○ FALSE

Solution: True. Incrementing the sequence number differently doesn't make it any easier for an off-path attacker to guess it, and if you're on-path or MITM, you can see everything anyway.

Q1.7 TRUE or FALSE: Consider a modified version of DHCP, where in the server offer step, the server signs its message and sends its public key along with the signed message. This version of DHCP is secure against the DHCP spoofing attack.

○ TRUE ● FALSE

Solution: False. The client has no way to verify the public key. An attacker could easily send their own malicious public key and use that to sign a spoofed response.

Q1.8 TRUE or FALSE: TCP is secure against a DoS attack by a man-in-the-middle (MITM) because TCP guarantees delivery and will re-send messages until they are delivered.

○ TRUE ● FALSE

Solution: False. The MITM could just keep dropping packets so that messages never arrive. Also, the MITM can inject a RST packet, which ends the connection.

Q1.9 TRUE or FALSE: RSA-TLS is still secure if we use publically known lottery numbers as the value of the premaster secret (PS).

○ TRUE ● FALSE

Solution: False. An on-path or MITM attacker would know R_b and R_s (sent in plaintext) as well as PS, which would allow them to generate the symmetric keys and decrypt everything.

Question 2 **(15 min)**

Q2.1 Alice clears all her network settings and broadcasts a DHCP discover message. What information should she expect to receive in the DHCP offer in response?

- ■ (A) DNS server
- ☐ (B) Source port
- ■ (C) Lease time
- ☐ (D) Premaster secret
- ■ (E) Gateway router
- ■ (F) IP address

Solution: The DHCP offer will include IP address, DNS server, gateway router, and how long the client can have these ("lease time"). The source port is determined by the user's machine. DHCP does not involve any premaster secret.

Q2.2 After receiving the DHCP offer, Alice tries connecting to `www.cutecats.com`, but instead of pictures of cats, the site she gets is filled with dog photos.

How did the attacker compromise DHCP to accomplish this?

Solution: Since the DHCP discover message is broadcasted, any local attacker can hear the host's request. The attacker then spoofed the DHCP response by racing the actual server to send the DHCP offer to the client.

Which of the following could the attacker have replaced?

- ■ (G) DNS server
- ☐ (H) Source port
- ☐ (I) Lease time
- ☐ (J) Premaster secret
- ■ (K) Gateway router
- ☐ (L) IP address

Solution: Replacing the DNS server allows the attacker to redirect address lookups to a machine of the attacker's choosing. Replacing the gateway router allows the attacker to intercept all of the host's off-subnet traffic.

Q2.3 Alice clears all her network settings and starts a new connection to `www.cutecats.com` with TCP. Now an off-path attacker wants to send a packet to the server to interfere with Alice's connection. What information do they need to know?

☐ (A) Server sequence number

■ (B) Source port

■ (C) Client sequence number

■ (D) Destination IP address

■ (E) Destination port

■ (F) Source IP address

Solution: An off-path attacker needs the IP addresses, ports, and sequence numbers to inject a packet. However, the server sequence number is not necessary because the server won't reject a packet with an incorrect ACK number.

Q2.4 At some point, Alice's connection with `www.cutecats.com` is suddenly terminated. Assuming some information was leaked and the attacker correctly guessed the fields from the previous part, how was the attacker able to execute this attack?

○ (G) —— ○ (H) —— ○ (I) —— ○ (J) —— ○ (K) —— ○ (L) ——

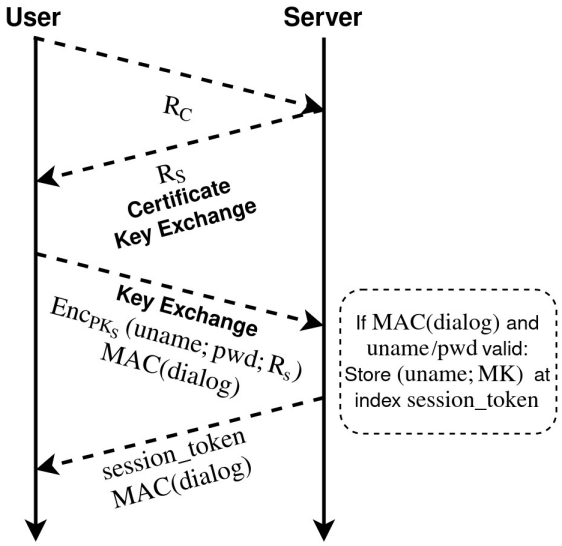
Solution: The attacker injected a RST packet with the correctly guessed fields. This terminates the TCP connection.

Question 3 **(37 min)**

FastCash is a fast banking service which requires users to log in before making a transfer, and uses TLS with ephemeral Diffie Hellman and RSA certificates to secure all their connections. They implemented a TLS extension called *0-Round Trip (0-RTT)* to speed up the connection process. 0-RTT changes the initial handshake as follows:

- Users authenticate themselves during the second round of the handshake
- If the user authenticates correctly, the server stores a session_token for that user

(Recall that in TLS, PS, R_S, *and* R_C *generate a master key set* MK *which contains all the symmetric keys.* Enc_{PK_S} *denotes RSA encryption using the server's public RSA key.)*



Simplified diagram of modified initial TLS handshake

A user only needs to perform the modified TLS handshake once. To send an HTTP request after the initial connection ends, a user encrypts it using the keys derived in the initial handshake and attaches the session_token. The server verifies that the entry session_token : (uname, MK) exists and, if so, decrypts and executes the request as the user uname using the keys derived from MK.

Assume that any on-path TCP injection attacks are impossible, and that once a user makes the initial modified TLS handshake, they will use the 0-RTT extension for future requests to the server.

Q3.1 An on-path attacker observes an initial TLS handshake between a user and server, as well as a subsequent 0-RTT packet which contains an encrypted HTTP request. What can they do?

☐ (A) Read the user's future communications

☐ (B) Pretend to be the server to the user

☐ (C) Pretend to be the user to the server in a new handshake

■ (D) Replay the encrypted HTTP request to the server

☐ (E) Learn the master key set

☐ (F) None of the above

Solution: The adversary can't learn any of the keys and so can't decrypt anything or fake being the server. While normally TLS doesn't authenticate the client, the 0-RTT extension involves authentication so without knowledge of the username/password the adversary can't pretend to be the user either. Including R_S in the encryption stops the ciphertext from being replayed in a different session.

The adversary knows the session_token, so they can use the 0-RTT extension to replay an encrypted HTTP request they observed.

Q3.2 Suppose we removed R_S from the user's KeyExchange in the third step of the handshake. After observing an initial handshake between a user and the server, what can an on-path adversary do?

☐ (G) Read the user's future communications

☐ (H) Pretend to be the server to the user

■ (I) Pretend to be the user to the server in a new handshake

☐ (J) Learn the premaster secret

☐ (K) Learn the master key set

☐ (L) None of the above

Solution: The adversary can't derive the premaster secret due to DH, and thus can't learn the master key set, violate forward secrecy, or learn future communications. Furthermore, they can't pretend to be the server to the user: the server's KeyExchange message, so the attacker can't modify or forge it, and consequently the adversary cannot predict the result of the Diffie-Hellman key exchange, the premaster secret, or the master key set.

However, the adversary can initiate a new handshake and replay the $\mathsf{Enc}_{\mathsf{PK}_S}(\mathsf{uname}; \mathsf{pwd})$ ciphertext observed in the first handshake to log in as the user.

Q3.3 Due to a bug, an on-path adversary is able to choose the server's R_S. After observing an initial handshake between a user and the server, what can they do?

☐ (A) Read the user's future communications

☐ (B) Pretend to be the server to the user

■ (C) Pretend to be the user to the server in a new handshake

☐ (D) Learn the premaster secret

☐ (E) Learn the master key set

☐ (F) None of the above

Solution: Same reasoning as above. The only thing that's different is the adversary has to force the server's R_S to be the same as used in the initial handshake to get the replay to work.

Q3.4 An on-path adversary observes a user and the server communicating using 0-RTT for some time (without observing the initial handshake). At some point in the future, the adversary manages to learn all of the server's session_token : (uname, MK) entries. What can they do?

■ (G) Read the user's future communications

■ (H) Pretend to be the server to the user

☐ (I) Pretend to be the user to the server in a new handshake

☐ (J) Learn the premaster secret

■ (K) Learn the master key set

☐ (L) None of the above

Solution: The adversary can learn the master key set (but not the premaster secret). This allows them to decrypt all future communications. Note that since we are essentially using a long-term private key (it is reused for all subsequent 0-RTT handshakes from the same user), we nullify the forward secrecy of using the Diffie-Hellman key exchange once the adversary has the master key set.

The adversary has no way to learn a valid ciphertext for the user's password so they can't pretend to be them. Since in future connections the user doesn't check the server's certificate, the adversary can pretend to be the server.

Q3.5 Consider a MITM adversary during the initial handshake between a user and the server. Describe how this adversary can send a malicious HTTP request that appears to come from the legitimate user (Be specific with what is sent). Disregard any bugs from previous parts.

Solution: The adversary should do a DH MITM. When the server sends $\mathbf{R_S}$, the adversary should relay that same value on to the client. When the client sends the

encrypted password, the adversary forces the client's connection to end by sending a RST packet. Next, the adversary replays this encrypted password to the server. This will be accepted by the server since $\mathsf{R_S}$ will be the same as the server was expecting, but replaces the client's g^b value with the adversary's own $g^{b'}$. The adversary can then compute the result of the DH key exchange with the server, derive the premaster secret and master key set, and uses the derived MAC key to finish the handshake with the server. The adversary can now log in as the user using the session token returned by the server.

Normally TLS does not authenticate the client, so a MITM can always take over a connection initiated by the client. The key difference here is that the 0-RTT extension effectively authenticates the user. So the MITM can not only take over the connection, but also does so authenticated as the client; the server thinks messages are coming from the user, when actually they are coming from the adversary.