Assignment 5 - Report - CS23MTECH11026

Q1. Assume that TLS handshake does not use any nonces. Explain how Trudy can be successful in launching session/connection replay attacks by capturing all the messages exchanged between Alice and Bob a while ago. You can assume Alice and Bob used TLS for securing their communication related to say ordering an item for e-commerce, online payments, secure file transfers, etc. Can Trudy replay Alice's previous messages with Bob for successfully launching a session/connection relay attack with Bob? Explain your answer. Can Trudy replay Bob's previous messages with Alice for successfully launching a session/connection replay attack? Explain your answer.

Ans: Yes, in the case of Trudy acting as Alice because, as mentioned in the question, the TLS handshake does not use any nonces, then Bob won't be able to differentiate between Alice and Trudy because, for Trudy also, Bob will share the same random number (Rs). Trudy will also generate the same Master Secret and will be successful in a replay attack with Bob. Whereas in the second case, Trudy as Bob, Trudy doesn't have Bob's private key to decrypt the PMS received form Alice as PMS is encrypted with Bob's public key. Thus, further communication will stop, and no handshake will happen. Hence, Trudy is unsuccessful in launching a replay attack with Alice.

Q2. Explain how nonces employed in TLS help in preventing session/connection replay attacks in Q1.

Ans: Referring to the answer of question 1. If nonces are employed then Bob will generate the different Random numbers (Rs') and will share the Rs' with Trudy but Trudy will discard those messages and use all the Alice ones. Now the Master Secrete at Trudy is MS (Alice one) but at Bob the Master Secrete generated is MS' because Bob used Rs'. Thus, employing nonces in TLS will prevent replay attacks due to the inequality of Master secret (MS != MS') which will eventually stop the Handshaking process.

Q3. How does Alice derive the PreMaster Secret (PMS) which she wants to send to Bob? Refer RFC 5246.

Ans: As RSA is used for the key exchange Alice will generate a 48-byte PreMaster Secret (46-byte Random number + 2-byte TLS version). Then Bob receives PreMaster Secret encrypted with his public key sent from Alice. Now only Bob will able to decrypt because Bob will only have his private key.

Q4. Why can't Bob derive PMS and share it with Alice?

Ans: As the certificate_request message from server to client (Bob to Alice) is optional then it is not always possible to have a certificate_request message in all the TLS sessions. If the server (Bob) won't request the certificate of the client (Alice), then server (Bob) will not have the client's (Alice's) public key to encrypt the PMS generated at the server (Bob). Hence, for generalization purpose, the servers (Bob) don't generate PMS and share it with clients (Alice)

Q5. Think of a scenario in which it's possible for Bob to derive PMS and share it to Alice. Refer TLS 1.2 handshake message protocol and explain how it can be extended (say, by adding new messages) to achieve this behavior.

Ans: Referring to Q4's answer, we can make the certificate_request message from Bob to Alice (Server to Client) mandatory in the 4 phases of Handshaking. Thus, Bob will be able to encrypt the PMS with Alice's public key which was shared in the certificate (Alice to Bob).

Q6. Note that MS is derived by feeding PMS and nonces of Alice and Bob as inputs to a PRF (that is known to all) by both Alice and Bob independently. Similarly, MS and nonces of Alice and Bob, and key_block size are fed as inputs to a PRF to derive key material which are split into MAC keys, session keys and IVs (IVs for AES-CBC only) by both Alice and Bob independently. To lessen the burden(!) on Bob out of her love for Bob, Alice said that she would generate MS from PMS and nonces of Alice and Bob and directly share the MS to Bob by encrypting it with Bob's RSA public key. Trudy captured messages exchanged between Alice and Bob in this modified handshake protocol. Do you think Trudy can succeed in launching session/connection replay attacks on Bob? Justify your answer.

Ans: No, Trudy won't succeed in launching replay attacks on Bob. Even if Alice generates the MS and shares it with Bob. The key material generated at Bob's side in the session of Bob - Alice, will be different from the session of Bob - Trudy due to the use of nonces.

Q7. More love from Alice. Extension to Q6. Alice said that she would generate key material from MS and nonces of Alice and Bob, and key_block size and share the key material directly to Bob by encrypting it with Bob's public key. Trudy captured messages exchanged between Alice and Bob in this modified handshake protocol. Do you think Trudy can succeed in launching session/connection replay attacks on Bob? Justify your answer.

Ans: No, Trudy will not succeed in launching replay attacks on Bob. In the end, when the finished message is sent, both Alice and Bob will hash over all the messages exchanged and signed with their private key. Then both of them will share it after encrypting with the symmetric key. Now, Bob will verify all the message exchanges with Alice and Trudy (Thinking it’s Alice). Bob will discover that the message exchange with Trudy differs from what was exchanged with Alice on his (Bob) side (using nonces).

Q8. Sequence number counter (initially set to 0) is used by Alice to input the current value of the sequence number counter while calculating MAC for inclusion into TLS records for integrity protection. Assume that Alice has been sending 10 TLS records carrying application data (each of size 500 Bytes) to Bob. Trudy being Woman-in-the-Middle between Alice and Bob, deletes record numbered 7th. She wants to fool TCP's in-sequence delivery mechanism so that the TCP receiver at Bob thinks everything is perfect and forwards the received TLS records to the TLS layer. How could she get away and pass through TCP checks? Hint: Trudy has to manipulate TCP segments numbered 8th, 9th and 10th. How?

Ans: Yes, Trudy can fool TCP’s in-sequence delivery mechanism by specifically changing the TCP header fields like checksum, ACK number, and sequence number. Changing the sequence number of records 8, 9, 10 to 7, 8, 9 (from Alice to Bob) and accordingly changing the checksum so that the TCP of Bob thinks it’s perfect. Also, Trudy has to send the fake ACK of the 7th record to Alice.

Q9. Having successfully fooled the TCP receiver of Bob in Q8, do you think Trudy can fool the TLS receiver of Bob? Explain.

Ans: No, Trudy can’t fool the TLS receiver of Bob because the Sequence number counter is used as a parameter for keyed MAC. Due to that Bob’s TLS receiver can verify that is there any modification or deletion done in sequence number or not.

Q10. Assume that Trudy captured application data messages exchanged between Alice and Bob using TLS 1.2. Alice is a web browser whereas Bob is a web server with Digital Certificate signed by a CA using RSA. After a year from this correspondence between Alice and Bob, Trudy hacked into the webserver and stole Bob's private key. Explain how Trudy can decrypt all of the old application data exchanged between Alice and Bob? This means there is no forward secrecy. It's indeed possible when TLS_RSA_WITH_AES_256_CBC_SHA256 is used as the cipher suite.

Ans: Considering That Trudy had captured all the messages that were exchanged between Alice and Bob a year ago which contains the ServerHello and ClientHello through which Trudy has access to the random number shared between Alice and Bob. Now as Trudy has hacked the webserver (Bob) and got Bob’s private key, Trudy can decrypt the PMS got from Alice. Then Trudy can generate MS using PMS, Rc, Rs and decrypt all the old application data exchanged between Alice and Bob.

Q11. You are tasked with providing perfect forward secrecy by fixing the issue described in Q10. What tweaks do you make to TLS_RSA_WITH_AES_256_CBC_SHA256 for that? Hint: You can't replace RSA with any other algorithm in the ciphersuite.

Ans: The issue described in Q10 can be simply avoided by creating a new DH private key for every handshake and utilising one of the DHE cypher suites which provides perfect forward secrecy. Changing the cipher suite's RSA to ECDH_RSA which provide's perfect forward secrecy.

Q12. Compare and contrast TLS_ECDH_RSA_WITH_AES_256_GCM_SHA and TLS_RSA_WITH_AES_256_CBC_SHA ciphersuites? Does TLS_ECDH_RSA_WITH_AES_256_GCM_SHA offer perfect forward secrecy? Explain.

Ans:
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA
- Uses Elliptic Curve Diffie-Hellman (ECDH) for key exchange
- RSA digital signature for key authentication.
- Uses AES in Galois/Counter Mode (GCM) for encryption and SHA for message authentication.
- Provides perfect forward secrecy because ECDH generates a new private key for each handshake.

TLS_RSA_WITH_AES_256_CBC_SHA
- Uses RSA for key exchange and authentication
- Uses AES in Cipher block chaining (CBC) for encryption and SHA for message authentication.
- Does not provide perfect forward secrecy.

Q13. Refer RFC 5246 on Cipher Suites of TLS 1.2 and list down the ones that offer perfect forward secrecy

Ans:
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

Q14. What measures are taken in TLS 1.2 with respect to TLS_ECDH_RSA_WITH_AES_256_CBC_SHA cipher suite to guard against various attacks?

Ans: Using ECDH_RSA for key exchange and authentication which provides perfect forward secrecy that prevents replay attacks. Also, AES_256_CBC ensures data confidentiality because of the involvement of the Initialization vector in the CBC method or adding randomness even though the plaintext are same. Due to SHA (secure hashing algorithm) the overall session guarantees integrity and authenticity.

Q15. Refer RFC 8446 on Cipher Suites of TLS 1.3 and list down the ones that offer perfect forward secrecy

Ans: TLS 1.3 is designed to prioritize Perfect Forward Secrecy (PFS). Thus, all the cipher used in TLS 1.3 provides PFS. Major key exchange methods used in cipher suites of TLS 1.3 are based on the DHE and ECDHE methods.
Examples: TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384, TLS_AES_128_CCM_SHA256

Q16. Privacy issues with TLS 1.2: Does any 3rd party like ISPs/enterprises profile their users (i.e., browsing patterns) even though their application data is encrypted? Explain!

Ans: Even though TLS 1.2 encrypts the content of data due to the availability of metadata privacy issues have been raised. Third-party like ISPs / enterprises can detect such browsing patterns, traffic patterns of users, DNS queries, destination IPs, etc. Thus, additional measures should be taken into consideration to minimize the availability of metadata.

References:

https://tools.ietf.org/html/rfc5246
https://www.coursera.org/learn/crypto/lecture/WZUsh/case-study-tls-1-2

PLAGIARISM STATEMENT

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honour violations by other students if I become aware of it.

Name: Bhargav Patel
Date: 1/2/24
Signature: B.P