**Assignment -5**

**Recall the topics covered on role of nonces (one each from Alice and Bob) and sequence number counters (one each for Alice and Bob) and answer the following queries in crisp (5-10 lines). Alice is a web browser whereas Bob is a web server with Digital Certificate signed by a root CA using RSA. Also assume that RSA is used for key exchange between Alice and Bob, unless stated otherwise.**

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:** Trudy will be successful in launching session/connection reply attacks as there is no randomness, the accumulated hash computed as a part of the finished message will be identical to both trudy and bob. Nonces are the one that guards against MITM attack by adding the flavor of randomness to the accumulated hash.

Yes, trudy would be successful in replaying Alice’s previous message as given that we are not using nonces the server (bob) won’t be able to detect that its replayed message and would consider it as Alice only.

No, Trudy won’t be successful in replaying Bob’s previous message and fooling Alice. As for the replayed message from the trudy, impersonating as bob would lead to generation of new PMS at the client's side which would lead to mismatch of hash in the finished phase, and client would detect the replay attack.

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

**Ans:** Nonces add randomness in the handshake mechanism. So if someone tries to replay the message the corresponding hash intended for verifying the integrity would not match during the final finished phase of the handshake. Resulting in detection of a reply/ MITM attack.

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

**Ans:** During server\_key\_exchange message the PMS is set up at the client. The client uses ephemeral Diffie-Hellman exponent or RSA. If later then it sends the 48-byte RSA encrypted PMS else it contains the client's Diffie-Hellman public value.

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

**Ans:** Bob can generate PMS but it requires him to request Alice’s certificate in certificate\_request of phase II of the handshake protocol.

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:** Bob needs to mandatory request the Alice certificate with Certificate\_request message to get her public key and which can later be used to encrypt the PMS.

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:** Trudy won't be successful in launching the replay attacks on bob as still nonces are intact in the handshake mechanism of TSL which by their randomness will easily detect any MITM attack i.e. for the replayed message the server (bob) would generate a new number (nonces) and this difference would lead to different hash and would mismatch during the Finished Phase. In case we want to detect replay attacks earlier before fourth phase then we also have to compute MS at the bob’s end and compare that with the received MS by decrypting it with bob’s secret key and surely there would be a mismatch.

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:** Trudy won't be successful in launching the replay attacks on bob as still nonces are intact in the handshake mechanism of TSL which by their randomness will easily detect any MITM attack. During the Finished Phase the Accumulated message hash that the server computed would be different from the received replayed hash message due to difference in the random number generated at the server’s end.

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, it can be made possible by adjusting some of the fields of the TCP header i.e. the sequence number, ack number, checksum and by sending some fake acknowledgements to hide the fakeness. The sequence number of 8th, 9th and 10th should be changed to 7th,8th and 9th respectively. Also as changed the sequence number the checksum of that segment will change so we should also change the checksum of that respective segment. We also need to change the Acknowledgement number in the respective packets with that of 7th, 8th and 9th packet’s acknowledgement number. Also Trudy should send a fake acknowledgment (ACK) to Alice, indicating that the 7th record has been received successfully.

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

**Ans:** As earlier mentioned we are using Sequence number counter as one of the parameters while calculating the segment keyed-MAC, and then thus incrementing the counter. If an attacker attempts to replay a TLS record with a previously used sequence number, the recipient can detect the discrepancy and reject the record, considering it a potential replay attack.

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:** Assuming that prior only trudy is capturing all the encrypted data exchanged between bob and alice. Also assume that trudy already has the random number of all the communication as the ClientHello and ServerHello are plaintext message exchange and can be known to all.

Now when Trudy dumped Bob's server and stole Bob’s private key then she can decrypt the PMS and can generate the key material and thus the session key. And mentioned earlier she already has the encrypted message and using this session key she can spy into all the previous messages.

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:** We can use Ephemeral Diffie-Hellman (DHE) key exchange alongside RSA for authentication. By using DHE, each session will negotiate a unique, ephemeral key for key exchange. Even if an attacker gains access to the long-term RSA private key later, it won't compromise the confidentiality of past communications since the session key was ephemeral. Here, the keypairs are Ephemeral (generated for each connection) rather than using the public/private key from the certificate.

Q12. Compare and contrast TLS\_ECDH\_RSA\_WITH\_AES\_256\_GCM\_SHA andTLS\_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:**

Key Exchange: Elliptic Curve Diffie-Hellman (ECDH)

The ECDH key exchange provides forward secrecy by generating an ephemeral key for each session.

Cipher: AES-GCM (Galois/Counter Mode)

Hash: SHA (Secure Hash Algorithm)

AES-GCM is an authenticated encryption mode providing confidentiality and integrity.

ECDH offers strong security with smaller key sizes compared to traditional RSA

**TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA:**

Key Exchange: RSA

RSA key exchange lacks perfect forward secrecy, as the same key is used for multiple sessions. Compromising the RSA private key can also decrypt past communications.

Cipher: AES-CBC (Cipher Block Chaining)

Hash: SHA

AES-CBC provides confidentiality but requires an additional HMAC for integrity.

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

**Ans:**

DHE (Ephemeral Diffie-Hellman) Cipher Suites:

TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA

TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA

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\_DSS\_WITH\_AES\_128\_CBC\_SHA256

TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA256

ECDHE (Ephemeral Elliptic Curve Diffie-Hellman) Cipher Suites:

TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA

TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA

TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256

TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384

TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA

TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA

TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256

TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384

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:** In TLS 1.2 with the TLS\_ECDH\_RSA\_WITH\_AES\_256\_CBC\_SHA cipher suite, security is enhanced through measures like Perfect Forward Secrecy (PFS) using Ephemeral Elliptic Curve Diffie-Hellman. Elliptic Curve Cryptography (ECC) is employed for strong security with shorter key lengths, enhancing performance. AES-256 encryption in Cipher Block Chaining (CBC) mode ensures data confidentiality, while HMAC-SHA1 provides integrity during transmission. The RSA digital signature in the key exchange process authenticates parties, guarding against Man-in-the-Middle attacks. The use of the Secure Hash Algorithm (SHA) for message digests further contributes to the overall integrity and authenticity of the communication.

Q14. Refer RFC 8446 on Cipher Suites of TLS 1.3 and list down the ones that offer perfect forward secrecy

**Ans:** TLS 1.3 prioritizes PFS, and as a result, all supported cipher suites inherently provide forward secrecy, enhancing the security of encrypted communications

Ephemeral Diffie-Hellman (DHE): TLS\_AES\_128\_GCM\_SHA256 (DHE), TLS\_AES\_256\_GCM\_SHA384 (DHE)

Ephemeral Elliptic Curve Diffie-Hellman (ECDHE): TLS\_AES\_128\_GCM\_SHA256 (ECDHE), TLS\_AES\_256\_GCM\_SHA384 (ECDHE)

Q15. 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:** Privacy concerns exist with TLS 1.2. While TLS encrypts the content of communication, third parties like ISPs and enterprises can analyze metadata, including server names and connection patterns, leading to potential user profiling. Encrypted application data alone may not protect against metadata analysis, raising privacy issues even when the content is secure. Upgrading to newer TLS versions, like TLS 1.3, can enhance privacy by minimizing metadata exposure during communication.

.

**References:**

* **Slide deck on TLS**
* <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: Yug Patel

Date: 21/01/2024

Signature: Yug Patel