CSE 4057 Programming Assignment

Due: 20.05.2020, 23:59

In this programming assignment you will implement a simple peer to peer (P2P) messaging system with several security features.

Prerequisite: It is assumed that you have main knowledge about socket programming and you can write multithreaded applications. If you didn't take Computer Networks or Operating Systems in the past semesters, please let me know.

In the messaging system, there should be a server which stores the certificated public keys. Public key of the server is known by every user (the server is considered as a trusted certificate authority).

The required security features are described as follows.

1. **Public Key Certification**. Each user should generate a public-private key pair once and register the server with her username and public key. Server signs this key and creates a certificate, stores the certificate and also sends a copy to the user. When user receives the certificate, she verifies that the certificate is correct and the public key is correctly received by the server.
2. **Handshaking.** When user1 wants to communicate with user2, she sends a hello message together with her public key certificate to user2. User2 sends back a random number (nonce) together with his public key certificate. User1 encrypts the nonce with her private key and sends back to user2. User2 verifies the nonce and send an acknowledgement to user1. Then user1 generates a master secret and sends it to user2 in a secure way. (Here we assume that users know each other's user names and contact addresses. However you can implement your code such that these information together with the public key certificates obtained from the server.)
3. **Key Generation.** Both user1 and user2 generates necessary keys for encryption and Message Authentication Code (MAC), as well as initialization vector(s) (IV). These keys and IV(s) should be derived from master secret (it would be preferable if you generate different keys for two directions).
4. **Message Encryption.** All the messages between pairs must be encrypted using a block cipher. You may use any standard block cipher you like, but all the messages must be encrypted using CBC mode (with the IV generated in the previous step).
5. **Integrity Check.** Every message going over the network should have a MAC, to enable detection of a malicious attacker tampering with the messages en route. You can either use hashing (HMAC) or block cipher (CMAC) to generate MAC. Clearly describe your choice.
6. **Resistance to Replay Attacks.** Even after you secure all the transmitted messages with encryption and MACs, there is still an obvious replay attack possible. Trudy can capture a message and repeatedly send this message. You should prevent the attacker from replaying any message. Clearly describe your method.
7. **(Bonus) Key Update.** Provide a key update mechanism to provide extra security, such that if any encryption key for any message is compromised, previous messages cannot be decrypted. Clearly describe your method.

8. (Bonus) **Sending end-to-end encrypted message to an offline user.** User1 would be able to send an encrypted message even if user2 is offline. User2 will be able to receive this message when it becomes online. For this purpose, you may need to modify the handshaking mechanism properly.
9. (Bonus) **Sending files.** User1 would be able to send a file (an image, a pdf, etc) to user2. The file should be encrypted with another key and stored somewhere in the server together with a digital signature (to provide authentication and integrity of the file). Then the necessary keys should be sent to user2 in a secure way. User2 would receive the file from the server, verifies the digital signature and decryptes it.

Print all the messages (plaintext or ciphertext) sent and received by any peer or the server to a log file.

Security Holes

Although you will implement many security features described above, there could stil exist some security holes. Try to identify security holes as much as possible and offer some solutions (You may implement additional features and get bonus points).

Additional Info

You can use any programming language of your choice. You may assume that there are just two users with known usernames and IP addresses. But, you are welcome to extend your study for more than two users which obtain the contact information of other users from the server (you may gain some additional bonus).

You may use libraries for cryptographic algorithms (AES, HMAC, RSA, etc) But, you should not use SSL libraries. You need to implement hanshaking mechanism in your own.

You may work **alone** or in a **group of two**. If you work in a group, you should clearly declare (i) how you communicate and coordinate during the current Covid-19 outbreak condition, (ii) which parts are done by whom (give detailed information about division of labor), (iii) and how did you merge your codes (again a detailed information please).

What to submit?

In addition to your well-commented code, you should submit **a README file containing your name and ID (or name and ID of group members), a detailed description of design choices you made in implementing the required security features and your comments and ideas about the potential security holes and countermeasures**. If you do part of the project, please clearly indicate which parts are implemented. Please submit all these files in a zip file (file name should include your name(s) and surname(s)) to cse457submit@gmail.com

Your codes will be checked for plagiarism. Any kind of plagiarism will be severely punished.