NS Assignment 2

Harish Fulara - 2014143

Encryption Used: AES with a 256 bit key.

Message Authentication Code Used: HMAC

The system works as follows:

Scenario: Alice wants to talk to Bob.

A - Alice
B – Bob
K_A – Shared key between A and KDC. It is derived from passphrase of A.
K_B – Shared key between B and KDC. It is derived from pasphrase of B.
K_{KDC} – Passphrase of KDC. Only KDC knows it.
TGT – Ticket Granting Ticket
K_{AB} – Session key for encrypting chat messages between Alice and Bob.

First, both Alice and Bob needs to be logged into the chat server before chatting with each other.

Logging into Chat Server

1. Alice first needs to login to the chat portal. Alice will send its login credentials (Username + SHA256(Passphrase)) to the KDC. As KDC also knows the passphrase of Alice, it will generate a SHA2256 digest of Alice's passphrase and will compare it with the digest sent by Alice. If the two digest matches, then Alice is OK to login.

2. Alice will then send K_A (Passphrase + Timestamp) to the KDC. The KDC decrypts the encrypted block and takes out Passphrase and Timestamp from it. The KDC will compare the timestamp with it's own time and if it is received within 5 minutes, then KDC will accept the block. The KDC will then verify the passphrase in the block against the passphrase in its database and will continue only if both the passphrase matches.

3. KDC will now generate a TGT = K_{KDC} (Alice's Username + Expiry Timestamp). The TGT contains Alice's username and expiry time (time after which the TGT is not valid) encrypted with passphrase of KDC. The KDC will send TGT to Alice.

4. Alice is successfully logged in when it receives the TGT.

5. Similarly, Bob will follow the above procedure to login to the chat portal.

Now Alice and Bob will negotiate the session key from KDC to encryt their chat messages.

Negotiating the key

1. Alice will send the TGT to the KDC.

2. KDC will verify that it is a valid TGT and is not expired. If OK, KDC will send the ACK to Alice.

3. Alice will now send the request for session key to chat with Bob to KDC. The request block will contain K_A (Bob's Username + Timestamp).

4. The KDC will decrypt the request block and take out Bob's username and Timestamp from it. The KDC will compare the timestamp with it's own time and if it is received within 5 minutes, then KDC will accept the request block.

5. The KDC will now generate a 256-bit session key K_{AB} that Bob and Alice can use to encrypt their chat messges. The KDC will send the response = K_A (K_B (Ticket for Bob) + K_{AB} + Expiry Timestamp) to Alice.

6. Alice will decrypt the response from KDC and take out K_B (Ticket for Bob), session key K_{AB} and Expiry Timestamp (time after which session key is invalid) from it. Alice will store the session key K_{AB} in its memory and will forward the ticket, i.e, K_B (Ticket for Bob) to Bob.

7. Bob will receive and decrypt the ticket. The ticket contains Alice's Username, session key K_{AB} and Expiry Timestamp (time after which session key is invalid). Bob will accept the ticket and store the session key K_{AB} in its memory only when Bob authenticates Alice.

Now Bob and Alice will authenticate each other before actual communication.

Two way authentication between Bob and Alice

1. Bob will send K_{AB} (Bob's Username + Nonce1) to Alice.

2. Since Alice also knows K_{AB}, so it will decrypt the block and take out Bob's username and Nonce1 from it. Alice will send K_{AB} (Alice's Username + Nonce1 + Nonce2) to Bob.

3. Since Bob also knows K_{AB}, so it will decrypt the block and take out Alice's username and Nonce1 and Nonce2 from it. If both the nonce matches (Nonce sent by Bob to Alice == Nonce sent by Alice to Bob == Nonce1), then Bob will authenticate Alice. Bob will send K_{AB} (Bob's Username + Nonce2) to Alice.

4. Since Alice also knows K_{AB}, so it will decrypt the block and take out Bob's username and Nonce2 from it. If both the nonce matches (Nonce sent by Alice to Bob == Nonce sent by Bob to Alice == Nonce2), then Alice will authenticate Bob.

5. Both Alice and Bob have authenticated each other. Now both Alice and Bob can chat with each other using the session key K_{AB} to encrypt their chat messages untill the session key expires.

How the system protects against Replay Attack

In our system, potential replay attacks can occur at two points:

1. When the client logs in to the chat server by sending its login credentials to the KDC, and

2. When the client sends the session key request to the KDC.

In the first case, some attacker can replay our authentication packet to the KDC and authenticate into the chat portal on our behalf.

In the second case, some attacker can replay our chat session request packet to the KDC and KDC will send the response packet containing the session key to the attacker.

To protect against replay attacks, the authentication protocol uses the concept of additional authentication data, such as the ticket lifetime, and most important, the client's timestamp.

In first case, KDC will check if the authentication packet’s timestamp is within five minutes of the time on KDC. If not, KDC will reject the packet. KDC will also reject the packet if the timestamp on the authentication packet is less than or equal to the last seen timestamp on the authentication packet sent by this client.

In the second case, KDC will check if the session key request packet’s timestamp is within five minutes of the time on KDC. If not, KDC will reject the packet. KDC will also reject the packet if the timestamp on the session key request packet is less than or equal to the last seen timestamp on the session key request packet sent by this client.

How the system protects against Reflection Attack

In our system, a potential replay attack can occur when Alice and Bob authenticate each other after negotiating the session key because they use **challenge-response** protocol for authentication.

Since both Alice and Bob include their identifier (their username) wihin their authentication response packets, they can prevent an attacker from doing a reflection because the attacker won’t be able to trick them into providing the answer to their own challenge.

You can find more information at: https://en.wikipedia.org/wiki/Reflection_attack