UNIT 8 SECURE SOCKET

LH - 4HRS

PRESENTED BY: ER. SHARAT MAHARJAN

NETWORK PROGRAMMING

PRIME COLLEGE, NAYABAZAAR

CONTENTS (LH - 4HRS)

8.1 Secure Communication

- Secure communication in Java is typically achieved using SSL/TLS.
- The javax.net.ssl package provides classes and interfaces for secure communication.
- Secure Communication involves ensuring that data transmitted over a network is encrypted and protected from eavesdropping or tampering.

8.2 Creating Secure Client Sockets

- Secure communication in Java is typically achieved using SSL/TLS.
- The javax.net.ssl package provides classes and interfaces for secure communication.
- Secure Communication involves ensuring that data transmitted over a network is encrypted and protected from eavesdropping or tampering.

Link: https://github.com/Sharatmaharjan/Np/blob/main/code/SecureSocketCommunication.java

8.3 Event Handlers

- Event handlers refer to handling various events such as connection establishment, data transmission, and disconnection. In Java, this can be managed using listeners and callback methods.

8.4 Session Management

- When dealing with secure communication over networks, especially using protocols like SSL/TLS (Secure Sockets Layer/Transport Layer Security), session management becomes crucial.
- Sessions in this context refer to the establishment of secure connections between a client and a server.

8.5 Client Mode

- It refers to the role and behavior of a software application acting as a client in establishing secure communications with a server over a network.
- In this mode, the client initiates the SSL/TLS handshake process by sending a ClientHello message to the server, proposing encryption algorithms, key exchange methods, and other parameters.
- The client then validates the server's digital certificate against a set of trusted Certificate Authorities (CAs) to ensure the server's identity and authenticity.
- Once mutual authentication and agreement on encryption parameters are achieved, a secure session is established.
- During this session, data exchanged between the client and server is encrypted to protect confidentiality and integrity.
- The client manages session resumption and termination as necessary.

8.6 Creating Secure Server Socket

- Creating a secure server socket involves setting up a server-side application that communicates over a secure channel using SSL/TLS protocols.
- Steps to Create a Secure Server Socket

Step 1: Load Keystore

- KeyStore: Load the server's KeyStore which contains its private key and certificate. This is used to prove the server’s identity during the SSL handshake.

Step 2: Initialize KeyManagerFactory

- KeyManagerFactory: Initialize with the KeyStore to provide authentication during the SSL handshake. It selects the appropriate key material for authentication.

Step 3: Initialize SSLContext

- SSLContext: Create an instance specifying the TLS protocol. Initialize it with KeyManagers from the KeyManagerFactory (and optionally TrustManagers from a TrustStore for client authentication).

Step 4: Create SSLServerSocketFactory

- SSLServerSocketFactory: Obtain from the SSLContext to create SSLServerSockets. It configures the sockets with the desired SSL parameters such as cipher suites and security protocols.

Step 5: Create SSLServerSocket

- SSLServerSocket: Create an instance using the SSLServerSocketFactory. Bind it to a specific port where the server will listen for incoming secure connections.

Step 6: Accept Client Connections

- Accept Loop: Continuously accept incoming connections from clients. Each client connection will result in an SSLSocket after the SSL handshake is successful.

Step 7: Handle Client Communication

- Client Handler: For each accepted connection, spawn a new thread or task to handle client communication. This typically involves reading from and writing to the SSLSocket.

Step 8: Close Resources

- Close: Properly close SSLSockets and SSLServerSocket when done to release resources.

Benefits of Using SSL/TLS

1. Encryption: Protects data exchanged between server and client from eavesdropping.
2. Authentication: Ensures both parties are who they claim to be.
3. Integrity: Guarantees that data cannot be tampered with during transmission.

8.7 Configure SSLServerSockets: Choosing the Cipher Suits, Session Management and Client Mode

Choosing Cipher Suites

- Cipher suites determine the encryption algorithms used for the secure connection. You can specify the cipher suites to be used by the SSLServerSocket.

Session Management

- Session management involves maintaining the state of the SSL session, which can improve performance by reusing sessions instead of negotiating new ones each time.

Client Mode

- SSLServerSocket can operate in both client and server mode. When configuring the server socket, it's typically set to server mode, but knowing how to set the mode can be useful in certain scenarios.

THANK YOU FOR YOUR ATTENTION