UNIT 8
SECURE SOCKET

LH - 4HRS

PRESENTED BY: **ER. SHARAT MAHARJAN**

NETWORK PROGRAMMING

PRIME COLLEGE, NAYABAZAAR

CONTENTS (LH - 4HRS)

8.1 Secure Communication

8.2 Creating Secure Client Sockets

8.3 Event Handlers

8.4 Session Management

8.5 Client Mode

8.6 Creating Secure Server Socket

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

8.1 Secure Communication

- Confidential communication through an open channel such as the public Internet absolutely requires that data be encrypted.
- The plain-text message is combined with the bits of the key according to a mathematical algorithm to produce the encrypted cipher- text.
- Using keys with more bits makes messages exponentially more difficult to decrypt by brute-force guessing of the key.
- In public key (or asymmetric) encryption, different keys are used to encrypt and decrypt the data. One key, called the public key, encrypts the data. This key can be given to anyone. A different key, called the private key, is used to decrypt the data. This must be kept secret but needs to be possessed by only one of the correspondents.

- JSSE allows you to create sockets and server sockets that transparently handle the negotiations and encryption necessary for secure communication.
- The **Java Secure Socket Extension** is divided into four packages:

javax.net.ssl The abstract classes that define Java's API for secure network communication.

javax.net The abstract socket factory classes used instead of constructors to create secure sockets.

java.security.cert The classes for handling the public-key certificates needed for SSL.

com.sun.net.ssl The concrete classes that implement the encryption algorithms and protocols in Sun's reference implementation of the JSSE.

8.2 Creating Secure Client Sockets

- Rather than constructing a java.net.Socket object with a constructor, you get one from a javax.net.ssl.SSLSocketFactory using its createSocket() method.
- SSLSocketFactory is an abstract class that follows the abstract factory design pattern.

SocketFactory factory = SSLSocketFactory.getDefault();

Socket socket = factory.createSocket("login.ibiblio.org", 7000);

- This either returns an instance of SSLSocketFactory or throws an InstantiationException if no concrete subclass can be found. Once you have a reference to the factory, use one of these five overloaded createSocket() methods to build an SSLSocket:

1. public abstract **Socket createSocket(String host, int port)** throws IOException, UnknownHostException
2. public abstract **Socket createSocket(InetAddress host, int port)** throws IOException
3. public abstract **Socket createSocket(String host, int port, InetAddress interface, int localPort)** throws IOException, UnknownHostException
4. public abstract **Socket createSocket(InetAddress host, int port, InetAddress interface, int localPort)** throws IOException, UnknownHostException
5. public abstract **Socket createSocket(Socket proxy, String host, int port, boolean autoClose)** throws IOException

8.3 Event Handlers

- Network communications are slow compared to the speed of most computers.
- Authenticated network communications are even slower. The necessary key generation and setup for a secure connection can easily take several seconds.
- JSSE uses the standard event model to notify programs when the handshaking between client and server is complete.
- In order to get notifications of handshake-complete events, simply implement the HandshakeCompletedListener interface:

public interface HandshakeCompletedListener extends java.util.EventListener

8.4 Session Management

- Web connections tend to be transitory ; every page requires a separate socket.
- For instance, checking out of Amazon.com on its secure server requires seven separate page loads, more if you have to edit an address or choose gift-wrapping.
- Because of the high overhead involved in handshaking between two hosts for secure communications, SSL allows sessions to be established that extend over multiple sockets.
- Different sockets within the same session use the same set of public and private keys.

8.5 Client Mode

- In most secure communications, the server is required to authenticate itself using the appropriate certificate.
- However, the client is not.
- That is, when I buy a book from Amazon using its secure server, it has to prove to my browser's satisfaction that it is indeed Amazon and not Random Hacker.
- However, I do not have to prove to Amazon that I am user.

8.6 Creating Secure Server Socket

- Secure client sockets are only half of the equation. The other half is SSL-enabled server sockets. These are instances of the javax.net.SSLServerSocket class:

public abstract class SSLServerSocket extends ServerSocket

- Like SSLSocket , all the constructors in this class are protected. Like SSLSocket , instances of SSLServerSocket are created by an abstract factory class, javax.net.SSLServerSocketFactory :

public abstract class SSLServerSocketFactory extends ServerSocketFactory

- Also like SSLSocketFactory , an instance of SSLServerSocketFactory is returned by a static SSLServerSocketFactory.getDefault( ) method:

public static ServerSocketFactory getDefault( )

- And like SSLSocketFactory, SSLServerSocketFactory has three overloaded createServerSocket( ) methods that return instances of SSLServerSocket:

public abstract ServerSocket createServerSocket(int port) throws IOException

public abstract ServerSocket createServerSocket(int port, int queueLength) throws IOException

public abstract ServerSocket createServerSocket(int port, int queueLength, InetAddress interface) throws IOException

8.7 Configure SSLServerSockets

- The SSLServerSocket class has the same three methods for determining which cipher suites are supported and enabled as SSLSocket does:

public abstract String[] getSupportedCipherSuites( )

public abstract String[] getEnabledCipherSuites( )

public abstract void setEnabledCipherSuites(String[] suites)

- Cipher suites are sets of instructions that enable secure network connections through Transport Layer Security (TLS), often still referred to as Secure Sockets Layer (SSL).
- Elliptic Curve Diffie–Hellman (ECDH), Elliptic Curve Digital Signature Algorithm (ECDSA) are some examples of cipher suites.

- Both client and server must agree to establish a session.
- The server side uses the setEnableSessionCreation( ) method to specify whether this will be allowed and the getEnableSessionCreation() method to determine whether this is currently allowed
- If the server disallows session creation, then a client that wants a session will still be able to connect. It just won't get a session and will have to handshake again for every socket.

- The SSLServerSocket class has two methods for determining and specifying whether client sockets are required to authenticate themselves to the server.
- By passing true to the setNeedClientAuth( ) method, you specify that only connections in which the client is able to authenticate itself will be accepted.
- By passing false , you specify that authentication is not required of clients.

THANK YOU FOR YOUR ATTENTION