Secure Sockets

Module java.base
Package javax.net.ssl

TLS (SSL) Overview

- Protocols to enables secure communication between network endpoints
  - Transport Layer Security (TLS)
  - Secure Socket Layer (SSL) - deprecated!
- Provides C and I (from the CIA requirements)
  - Confidentiality - data sent is encrypted
  - Integrity - ensures data is not altered
  - Availability - TLS doesn’t really protect against Denial-of-Service
- TLS 1.0 and TLS 1.1 are to be deprecated in 2020

Converting EchoClient to SecureClient

EchoClient.java

```
int portNumber = Integer.parseInt(args[1]);

try {
    Socket echoSocket = new Socket(hostName, portNumber);
```

SecureClient.java

```
int portNumber = Integer.parseInt(args[1]);

SSLSocketFactory factory =
        (SSLSocketFactory)SSLSocketFactory.getDefault();
try {
    SSLSocket socket = (SSLSocket)factory.createSocket(
            hostName, portNumber);
```

Converting from EchoServer

EchoServer.java

```
    int portNumber = Integer.parseInt(args[0]);

    try {
        ServerSocket serverSocket = new ServerSocket(portNumber);

        System.out.println("The server is listening at: " +
```

Converting to SecureServer

```
int portNumber = Integer.parseInt(args[0]);

SSLServerSocketFactory factory =
        (SSLServerSocketFactory)SSLServerSocketFactory.getDefault();
try {
    SSLServerSocket serverSocket =
            (SSLServerSocket)factory.createServerSocket(portNumber);

    System.out.println("The server is listening at: " +
        serverSocket.getInetAddress() + " on port " +
        serverSocket.getLocalPort());

    SSLSocket clientSocket = (SSLSocket)serverSocket.accept();
```

Generating a keystore/ truststore

- Caution: Avoid using self-signed keystores or certificates in production environments.

$ keytool -genkey -alias selfsigned -keystore mystore

- Respond to the prompts
  - Please use password ‘password’ when doing the exercise on last slide

Run the server and client

```
$ java -cp build -Djavax.net.ssl.keyStore=mystore  \
-Djavax.net.ssl.keyStorePassword=[your password]  \
SecureServer 4444
```

```
$ java -cp build -Djavax.net.ssl.trustStore=mystore  \
-Djavax.net.ss.trustStorePassword=[your password]  \
SecureClient localhost 4444
```

In-class Exercise

- Use the echodemo to create secure versions of the client and server
  - Copy EchoServer.java to SecureServer.java
  - Copy EchoClient.java to SecureClient.java
  - Modify the Secure*.java files to use Secure Sockets
- Create a self-signed keystore with password “password”
- Verify that your secure client and server can connect and exchange messages
- Tar/zip your updated echodemo project
- Email to me for grading