NAME: GVS SAI MADHAV
REG.NO: 19BCN7228

SSL:

SSLSocketClientWithTunneling :

```
import java.net.*;
import java.io.*;
import javax.net.ssl.*;
public class SSLSocketClientWithTunneling {
public static void main(String[] args) throws Exception {
new SSLSocketClientWithTunneling().doIt("www.verisign.com", 443);
}
String tunnelHost;
int tunnelPort;
public void doIt(String host, int port) {
try {
SSLSocketFactory factory =
(SSLSocketFactory)SSLSocketFactory.getDefault();
tunnelHost = System.getProperty("https.proxyHost");
tunnelPort = Integer.getInteger("https.proxyPort").intValue();
Socket tunnel = new Socket(tunnelHost, tunnelPort);
doTunnelHandshake(tunnel, host, port);
SSLSocket socket =
(SSLSocket)factory.createSocket(tunnel, host, port, true);
socket.addHandshakeCompletedListener(
new HandshakeCompletedListener() {
public void handshakeCompleted(
HandshakeCompletedEvent event) {
System.out.println("Handshake finished!");
System.out.println(
"\t CipherSuite:" + event.getCipherSuite());
System.out.println(
"\t SessionId " + event.getSession());
System.out.println(
"\t PeerHost " + event.getSession().getPeerHost());
}
}
);
socket.startHandshake();
PrintWriter out = new PrintWriter(
new BufferedWriter(
new OutputStreamWriter(
socket.getOutputStream())));
```

```
out.println("GET / HTTP/1.0");
out.println();
out.flush();
/*
* Make sure there were no surprises
*/
if (out.checkError())
System.out.println(
"SSLSocketClient: java.io.PrintWriter error");
/* read response */
BufferedReader in = new BufferedReader(
new InputStreamReader(
socket.getInputStream()));
String inputLine;
while ((inputLine = in.readLine()) != null)
System.out.println(inputLine);
in.close();
out.close();
socket.close();
tunnel.close();
} catch (Exception e) {
e.printStackTrace();
}
}
private void doTunnelHandshake(Socket tunnel, String host, int port)
throws IOException
{
OutputStream out = tunnel.getOutputStream();
String msg = "CONNECT " + host + ":" + port + " HTTP/1.0\n"
+ "User-Agent: "
+ sun.net.www.protocol.http.HttpURLConnection.userAgent
+ "\r\n\r\n";
byte b[];
try {
b = msg.getBytes("ASCII7");
} catch (UnsupportedEncodingException ignored) {
b = msg.getBytes();
}
out.write(b);
out.flush();
byte reply[] = new byte[200];
int replyLen = 0;
int newlinesSeen = 0;
boolean headerDone = false; /* Done on first newline */
```

```
InputStream in = tunnel.getInputStream();
boolean error = false;
while (newlinesSeen < 2) {
int i = in.read();
if (i < 0) {
throw new IOException("Unexpected EOF from proxy");
}
if (i == '\n') {
headerDone = true;
++newlinesSeen;
} else if (i != '\r') {
newlinesSeen = 0;
if (!headerDone && replyLen < reply.length) {
reply[replyLen++] = (byte) i;
}
}
}
String replyStr;
try {
replyStr = new String(reply, 0, replyLen, "ASCII7");
} catch (UnsupportedEncodingException ignored) {
replyStr = new String(reply, 0, replyLen);
}
if (!replyStr.startsWith("HTTP/1.0 200")) {
throw new IOException("Unable to tunnel through "
+ tunnelHost + ":" + tunnelPort
+ ". Proxy returns \"" + replyStr + "\"");
}
}
}
```

SSLSocketClientWithClientAuth :

```
import java.net.*;
import java.io.*;
import javax.net.ssl.*;
import javax.security.cert.X509Certificate;
import java.security.KeyStore;
public class SSLSocketClientWithClientAuth {
public static void main(String[] args) throws Exception {
String host = null;
int port = -1;
String path = null;
for (int i = 0; i < args.length; i++)
System.out.println(args[i]);
if (args.length < 3) {
System.out.println(
"USAGE: java SSLSocketClientWithClientAuth " +
"host port requestedfilepath");
System.exit(-1);
}
try {
host = args[0];
port = Integer.parseInt(args[1]);
path = args[2];
} catch (IllegalArgumentException e) {
System.out.println("USAGE: java SSLSocketClientWithClientAuth " +
"host port requestedfilepath");
System.exit(-1);
}
try {
/*
* Set up a key manager for client authentication
* if asked by the server. Use the implementation's
* default TrustStore and secureRandom routines.
*/
SSLSocketFactory factory = null;
try {
SSLContext ctx;
KeyManagerFactory kmf;
KeyStore ks;
char[] passphrase = "passphrase".toCharArray();
ctx = SSLContext.getInstance("TLS");
kmf = KeyManagerFactory.getInstance("SunX509");
ks = KeyStore.getInstance("JKS");
```

```
ks.load(new FileInputStream("testkeys"), passphrase);
kmf.init(ks, passphrase);
ctx.init(kmf.getKeyManagers(), null, null);
factory = ctx.getSocketFactory();
} catch (Exception e) {
throw new IOException(e.getMessage());
}
SSLSocket socket = (SSLSocket)factory.createSocket(host, port);
socket.startHandshake();
PrintWriter out = new PrintWriter(
new BufferedWriter(
new OutputStreamWriter(
socket.getOutputStream())));
out.println("GET " + path + " HTTP/1.0");
out.println();
out.flush();
if (out.checkError())
System.out.println(
"SSLSocketClient: java.io.PrintWriter error");
BufferedReader in = new BufferedReader(
new InputStreamReader(
socket.getInputStream()));
String inputLine;
while ((inputLine = in.readLine()) != null)
System.out.println(inputLine);
in.close();
out.close();
socket.close();
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

SSLSocketClientWithTunneling :

```
import java.net.*;
import java.io.*;
import javax.net.ssl.*;
public class SSLSocketClientWithTunneling {
public static void main(String[] args) throws Exception {
new SSLSocketClientWithTunneling().doIt("www.verisign.com", 443);
}
String tunnelHost;
int tunnelPort;
public void doIt(String host, int port) {
try {
SSLSocketFactory factory =
(SSLSocketFactory)SSLSocketFactory.getDefault();
tunnelHost = System.getProperty("https.proxyHost");
tunnelPort = Integer.getInteger("https.proxyPort").intValue();
Socket tunnel = new Socket(tunnelHost, tunnelPort);
doTunnelHandshake(tunnel, host, port);
SSLSocket socket =
(SSLSocket)factory.createSocket(tunnel, host, port, true);
socket.addHandshakeCompletedListener(
new HandshakeCompletedListener() {
public void handshakeCompleted(
HandshakeCompletedEvent event) {
System.out.println("Handshake finished!");
System.out.println(
"\t CipherSuite:" + event.getCipherSuite());
System.out.println(
"\t SessionId " + event.getSession());
System.out.println(
"\t PeerHost " + event.getSession().getPeerHost());
}
}
);
socket.startHandshake();
PrintWriter out = new PrintWriter(
new BufferedWriter(
new OutputStreamWriter(
socket.getOutputStream())));
out.println("GET / HTTP/1.0");
out.println();
out.flush();
/*
```

```
* Make sure there were no surprises
*/
if (out.checkError())
System.out.println(
"SSLSocketClient: java.io.PrintWriter error");
/* read response */
BufferedReader in = new BufferedReader(
new InputStreamReader(
socket.getInputStream()));
String inputLine;
while ((inputLine = in.readLine()) != null)
System.out.println(inputLine);
in.close();
out.close();
socket.close();
tunnel.close();
} catch (Exception e) {
e.printStackTrace();
}
}
private void doTunnelHandshake(Socket tunnel, String host, int port)
throws IOException
{
OutputStream out = tunnel.getOutputStream();
String msg = "CONNECT " + host + ":" + port + " HTTP/1.0\n"
+ "User-Agent: "
+ sun.net.www.protocol.http.HttpURLConnection.userAgent
+ "\r\n\r\n";
byte b[];
try {
b = msg.getBytes("ASCII7");
} catch (UnsupportedEncodingException ignored) {
b = msg.getBytes();
}
out.write(b);
out.flush();
byte reply[] = new byte[200];
int replyLen = 0;
int newlinesSeen = 0;
boolean headerDone = false; /* Done on first newline */
InputStream in = tunnel.getInputStream();
boolean error = false;
while (newlinesSeen < 2) {
int i = in.read();
```

```
if (i < 0) {
throw new IOException("Unexpected EOF from proxy");
}
if (i == '\n') {
headerDone = true;
++newlinesSeen;
} else if (i != '\r') {
newlinesSeen = 0;
if (!headerDone && replyLen < reply.length) {
reply[replyLen++] = (byte) i;
}
}
}
String replyStr;
try {
replyStr = new String(reply, 0, replyLen, "ASCII7");
} catch (UnsupportedEncodingException ignored) {
replyStr = new String(reply, 0, replyLen);
}
if (!replyStr.startsWith("HTTP/1.0 200")) {
throw new IOException("Unable to tunnel through "
+ tunnelHost + ":" + tunnelPort
+ ". Proxy returns \"" + replyStr + "\"");
}
}
}
```

```
HTTP/1.1 400 Bad Request
Date: Sat, 13 Nov 2021 02:52:05 GMT
Server: Apache
Content-Length: 226
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
</body></html>
```