

COMPUTING SUBJECT:	Secure Socket layer
TYPE:	Assignment
IDENTIFICATION:	Secure Socket
COPYRIGHT:	<i>Michael Claudius</i>
LEVEL:	Medium
TIME CONSUMPTION:	2-4 hours
EXTENT:	50 lines
OBJECTIVE:	SSL sockets in practice
PRECONDITIONS:	X509 No. 2, socket exercises Computer Networking Ch. 8.5
COMMANDS:	

IDENTIFICATION: SecureSocket/MC

The Mission

You are to make a secure connection communication by setting up a server and a client using the secure socket layer (SSL) by sharing the certificate provided by the server. This we shall do in three steps/assignments:

1. CertificateX509, Install Windows SDK and investigate the tools *makecert* and *pvk2pfx*
2. CreateCertificateX509, Create self-signed X509 Root and Server SSL certificates
3. **Secure SocketsC#, Use the certificates and SslStream for secure socket communication**

You have already done the first two assignments and this assignment is the Assignment No.3.

Purpose

The purpose of this assignment is to use the previously created certificates in server and client programs with SslStream class to achieve secure socket communication.

When surfing on the net it is easy to find many descriptions more or less useful, and in more or less updated versions. Here are some:

Useful links for for C#-programs with SSL

[http://msdn.microsoft.com/en-us/library/system.net.security.sslstream\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.net.security.sslstream(v=vs.110).aspx)

[http://msdn.microsoft.com/en-us/library/ms145056\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/ms145056(v=vs.110).aspx)

1. Local server and client on one computer

First clone the project from GitHub <https://github.com/rf20da2c3-3c/ClassDemo3CEchoApp> with the normal TCPEchoServer & TCPEchoClient or you can use a similar program or copy of your own (echo) project.

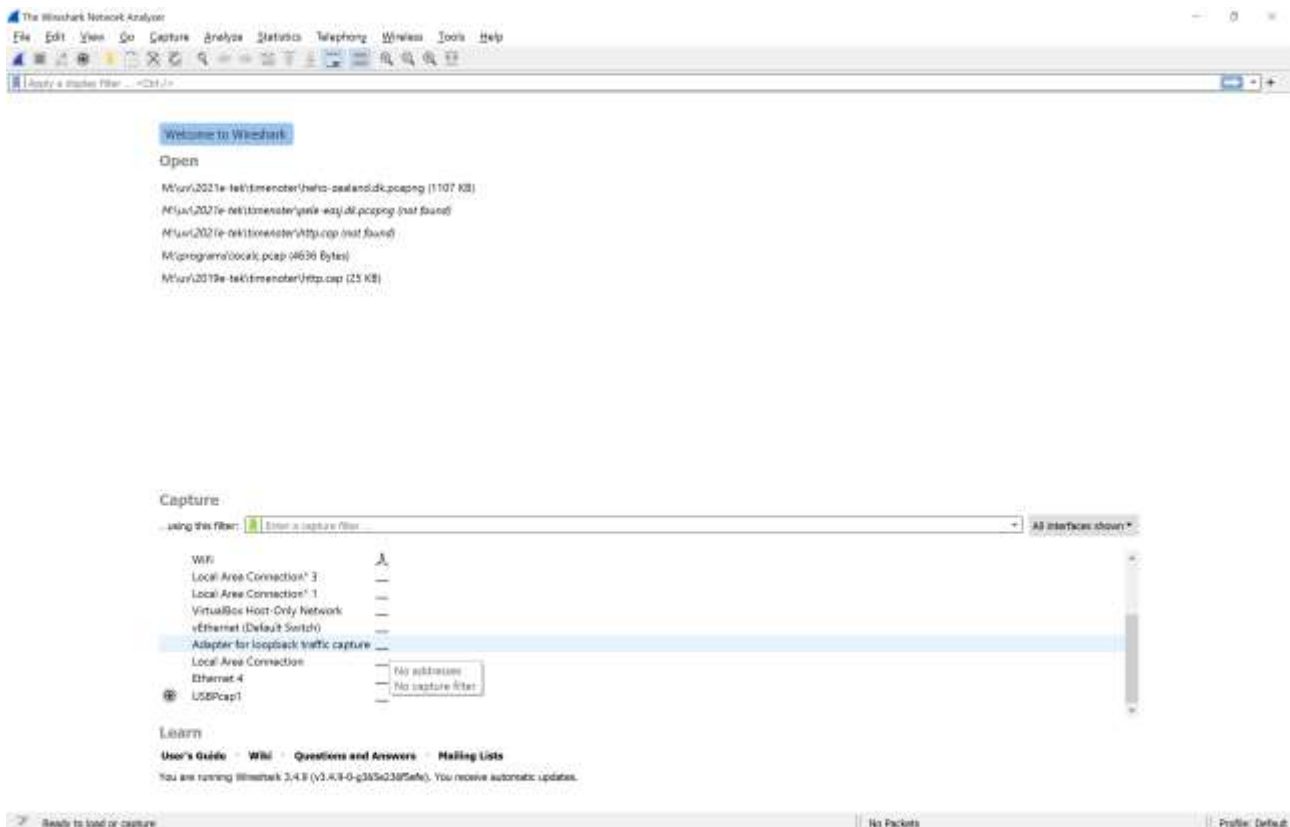
Run the TCPEchoServer and TCPEchoClient.

Type something in the client Window.

Test the communication.

2. Start wireshark to track / sniff the traffic

Start Wireshark and choose 'Adapter for loopback traffic adapter':



When you have run the echo-Server-Client

Stop wireshark and investigate the traffic packages (e.g. by use the filter 'tcp.port == 7')

Can you see your messages you have sent over the 'network'?

Extra1. Server and client on two different computers

Small group of students on the wireless local DMU LAN network ('mgv2-dmu2' password lanmagle) best performs this assignment.

Lookup the IP address of the server computer. (eg. Click: start -> run -> cmd -> 'type ipconfig').

Start the EchoServer on one computer.

On another computer modify the TcpEchoClient to create a socket to the server (your partner) ie. change "localhost" to the IP-address of the server.

Start the WireShark (a third partner) (i.e. start -> capture -> interface -> open)

Run the client and start to communicate.

Stop WireShark and view the captured packets.

You should now be able to read the packet content given as plain text.

You are now to change the echo-application to use Secure Socket Layer (SSL).

3. Use of SSL stream in the server on one computer

You can be inspired by the link:

[http://msdn.microsoft.com/en-us/library/system.net.security.sslstream\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.net.security.sslstream(v=vs.110).aspx)

Or just use the instructions given below.

Modify the TCPEchoServer to utilize a secure stream instead of the normal network stream (named: ns, unsecureStream or something like this) by extending the main().

In the start of main add the following definitions

```
string serverCertificateFile = "c:/certificates/ServerSSL.pfx";
bool clientCertificateRequired = false;
bool checkCertificateRevocation = true;
SslProtocols enabledSSLProtocols = SslProtocols.Tls;

X509Certificate serverCertificate =
    new X509Certificate2(serverCertificateFile, "mysecret");
```

Be sure you understand the classes SslProtocols and X509Certificate. Investigate !

Then change/extend the program with the following declarations:

```
Stream unsecureStream = connectionSocket.GetStream();
bool leaveInnerStreamOpen = false;
SslStream sslStream = new SslStream(unsecureStream, leaveInnerStreamOpen);
sslStream.AuthenticateAsServer(serverCertificate, clientCertificateRequired,
    enabledSSLProtocols, checkCertificateRevocation);

//Alternatively just
//sslStream.AuthenticateAsServer(serverCertificate)
```

Be sure you understand the classes SslStream and the method *AuthenticateAsServer*. Investigate !

Finally change the program to use sslStream instead of unsecureStream in the rest of the program. Compile and run. Then run the client.

This is of course not working as the client is not using SSL as required by the server.

4. Use of SSL stream in the client on one computer

You can be inspired by the link:

[http://msdn.microsoft.com/en-us/library/ms145056\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/ms145056(v=vs.110).aspx)

Or just use the instructions given below.

Modify the TCPEchoClient to utilize a secure stream instead of the normal network stream (named: ns, unsecureStream or something like this) by extending the main().

In the start of main add the following definitions:

```
bool leaveInnerStreamOpen = false;
SslStream sslStream = new SslStream(unsecureStream, leaveInnerStreamOpen);
sslStream.AuthenticateAsClient(certificateServerName);
```

where the certificateServerName is the subject certificate name you used when creating certificates like FakeServerName or whatever you choose.

Remember to utilize sslStream in the rest of the program.

Run the server and client again.

Now try to see the packages (using Wireshark), this time – what do you see?

Extra2. Secure server and client on two different computers

Follow the procedure in Assignment 2 but now for the SSL based TCPEchoServer and TCPEchoclient.

Remember that the sender and receiver must trust each other; i.e. one must install the others Root and server certificate.

Hopefully you now cannot see the plain text but only cipher text!