CSCI 4273/5273: Network Systems
Fall 2014
Programming Assignment 2

Due Date: 10/17/2014

Goal: The goal of this assignment is to incorporate security features in the communication between echo clients and echo server in the Echo system that was provided to you as part of the socket handout. In the process, you will learn secure network application programming using SSL (Secure Socket Layer).

Grade: 7% of your final grade is allocated for this assignment.

Recall that the Echo system allows multiple (echo) clients to connect concurrently to an echo server. After connecting, clients can send messages to the server and the server simply echoes them back. Your task is to incorporate the following security features in this system:

1. Clients authenticate the echo server before sending/receiving messages
2. Communication between the client and the server is encrypted using a symmetric key cryptographic algorithm
3. A new symmetric key is generated every time a client connects with the server
4. Both clients and server verify the integrity of the messages they receive

Use OpenSSL to incorporate these security features. The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. OpenSSL is already installed on CSEL servers. If you plan to use your own device to work on this assignment, please install OpenSSL first.

Certificate Authority Setup

SSL uses digital certificates for authentication. The server needs to get a signed public key certificate from a well-known and trusted Certificate Authority (e.g. VeriSign, GlobalSign, GoDaddy, etc.) that is trusted by the clients. However for this assignment, you (the student) will be considered a well-known and trusted authority (only within the domain of this assignment of course) and will provide a digital certificate for the server. To do this, you will need to generate:

- CA certificate: This is the CA's public certificate that's used to validate security certificates that have been signed by the CA. The file corresponding to this certificate will be called cacert.pem (in ./demoCA directory)
- CA private key: This is the private key corresponding to the public key in the CA certificate (in ./demoCA/private directory).

In order to generate these two files, follow these steps:

1. If you haven't already done so, make sure you install OpenSSL command line. Note that you may want to reinstall OpenSSL to include the API. http://www.openssl.org/source/ Note: your OpenSSL configuration (see openssl.cnf file in your OpenSSL Library) file will default your CA directory to ./demoCA
2. Create a temporary directory, cd into it, then type the following:
   - mkdir ./demoCA; mkdir ./demoCA/newcerts; mkdir ./demoCA/private
   - echo "01" > ./demoCA/serial; touch ./demoCA/index.txt;
   - openssl req -new -x509 -keyout ./demoCA/private/cakey.pem -out ./demoCA/cacert.pem

You will be prompted for several pieces of information, such as state, county, your name, organization, and so on. Feel free to fill them in with whatever you please, but to make more concrete, I would recommend using real information. You will also be asked to give a passphrase, which is used whenever you wish to sign a public key certificate. For the sake of this assignment, please set this to "netsys_2014" (all lower case). If you choose not to use this passphrase then I cannot help you create/debug signing certificates.

Congratulations! You have now created your CA's private key (cakey.pem: see in ./demoCA/private directory) and public certificate (cacert.pem: see in ./demoCA directory), thereby making you a Certified Authority! Both of these files must be included in your submission in order to verify certificates. Since, the echo clients will authenticate the echo server, make sure that these files are placed in the same directory as the echoClient.c.

Server Digital Certificate

To get a digital certificate, the server must first create public/private keys and then have you (CA) create a digital certificate certifying the public key. To generate a public/private key pair for the server, follow the steps below. The public key will be packed inside what is known as a "Certificate Signing Request" (CSR), which is a special type of file that is given to the CA to generate a digital certificate. Once the CA signs it, it becomes a full-blown Security Certificate, which can be verified using the CA certificate (cacert.pem). In order to generate public/private key pair, type the following command on your terminal:

```
openssl req –nodes –new –keyout server_priv.key –out server_cert_req.csr –outform PEM –keyform
PEM
```

No challenge password or optional company name are needed but your location must match the CA location because we are self-signing the certificates. Enter through these prompts. This generates a CSR (server_cert_req.csr) and the corresponding private key (server_priv.key).

As a CA, you can now sign the server's public key with your private key, turning it into a Security Certificate. To sign the CSR, type the following on the terminal:

```
openssl ca –in server_cert_req.csr –out server.cert
```

And there you have it. Sever.crt is the CA signed digital certificate for your server.

You should put these files with the source code of your project so you don't lose them (you will be turning them in with your assignment). NOTE: DO NOT COPY OTHER STUDENTS private keys/security certificates/public keys. This will be considered plagiarism.

If you want to know more about what you just did (or set more configuration options – there is a lot of information you can attach to a CA certificate), you can look at some of these websites:

http://www.freebsdmadeeasy.com/tutorials/freebsd/create-a-ca-with-openssl.php

Quick OpenSSL command line reference: http://wiki.samat.org/CheatSheet/OpenSSL

Incorporate SSL in Echo

Now that you have generated private keys and certificates including a certificate for the CA, lets put them to use. After creating your TCP socket all communication should be conducted through SSL/TLS. Before communication starts you must verify the certificate provided by the remote host. If verification fails then you should not communicate with that machine. After verification all communication must pass through the SSL (Secure Socket Layer). The following documents provided by HP are very extensive. I strongly recommend reviewing them, especially the SSL Programming Tutorial.

http://h71000.www7.hp.com/doc/83final/ba554_90007/ch04.html

Some Additional Resources

http://h71000.www7.hp.com/doc/83final/ba554_90007/ch04s03.html#sslhandshakesect
http://www.ibm.com/developerworks/linux/library/lopenssl/index.html
http://www.openssl.org/docs/
http://info.ssl.com/article.aspx?id=10241
https://www.globalsign.com/ssl-information-center/what-is-ssl.html —> check the video as well.

Assignment Submission

1. Submission deadline is Friday, October 17, midnight. No late submissions will be allowed, unless there is a valid excuse.
2. Submit a single zip file via the submission link on Moodle. Your zip file must contain echoClient.c, echoServer.c, cacert.pem, cakey.pem, server.cert, server_priv.key, Makefile and README. Include any other files that are needed for compiling and running your program. In your Makefile, use the following commands to compile your programs:
   ```
   gcc –o echoClient echoClient.c –lcrypto –lssl
   gcc –o echoServer echoServer.c –lcrypto –lssl
   ```
3. DO NOT include any object files in your submission.
4. In the README file, provide the following information: Your name; instructions on how to compile and run your program; current status of your program: whether it compiles or not, known bugs/limitations/unusual features, what parts of the program work, etc.; any other information that will be useful in grading your program.