COMS 4180 Network Security Spring 2018 Programming Assignment 2

Due Wednesday February 28, 2018 10:00pm Eastern time.

- This assignment is to be done individually.
- NO LATE SUBMISSIONS WILL BE ACCEPTED.
- The code for the programming problem must compile and run from the command line in Ubuntu 16.x or 17.x on the Google cloud account. Your code must be executable from the command line. DO NOT require that your code has to be run using a specific IDE, such as Eclipse.
- **Your code must be commented. Uncommented and poorly commented code will lose points.**

Submission Instructions (refer to the list of files at end of this document for what to include in your submission.)

- Assignments will be submitted via Courseworks.
- Submit a zip file containing a tar file with your submission
- The zip file name must be of the form UNI_#.<extension>, where "UNI" is your UNI and "#" is the number of the assignment, and the extension is zip or tgz.
- Please include your name at the top of each source code file and README file submitted.

Programming Problem: (150 points total)

A client will connect to a server using TLS (version 1.2) and send a file to the server. The connection will use 2-way authentication (both the client and server have certificates and exchange them). Use an existing library for the TLS sockets (i.e. openssl for C/C++, a Python wrapper for openssl, JAVA sslsocket). You must create certificates for the client and server (use openssl or JAVA keytool). Use ECDSA with a key length >= 256 for the public key algorithm and SHA256 for the hash function when creating certificates. The curve name for ECDSA is given as input when creating the certificate – refer to the documentation for openssl or keytool for the names. Use self-signed certs (you do not need to have a real CA to sign the certificates).

The client and server cannot have access to each other's private keys – the client's private key should only be on the client and the server's private key should only be on the server.

You may install any programming languages, their associated dependencies and useful shell commands on your VMs.

Note: Failure to perform 2-way authentication will result in a loss of 50 points (even if everything else works).

The default setting in the socket libraries is 1-way authentication.

Capture the traffic between the client and server using Wireshark or tcpdump. Have the client send the server a small file of < 1000 bytes when capturing traffic. Save the exchange to a pcap file. The pcap file will be included in your submission.

Client:

The client connects to a server using TLS with 2-way authentication then sends a file to the server (you may assume the file is < 1MB). The client disconnects from the server after sending the file to the server. The client takes the following information as command line inputs.

- Name of file to be sent to the server: Clearly indicate in your README file if the path of the file provided as input must be the full path or relevant to the directory containing the executable. You may require that the file be in the same directory as the executable.
- IP address and port number on which to contact the server
- certificate/keystore filenames

Server:
The server will start and wait for the client to connect. The server will save the file it receives from the client to the directory from which the server is executed. The connection should be torn down nicely and print a message to the terminal that the client has disconnected (no core dumps and no exceptions printed to the terminal). The server takes the following information as command line inputs.

- The port number on which to listen for a connection from the client.
- certificate/keystore filenames

Error Handling:
Programs will be tested for handling of invalid/garbage/missing input. The programs must check the validity of the input parameters and exit nicely if anything is invalid, printing a message specifying the required input parameters before exiting. This includes but is not limited to missing parameters, improper values (length, type, value), out of order parameters. Any runtime error must also be handled by printing an appropriate message and exiting nicely. (for example, segmentation faults in C/C++ will result in a grade of 0). NOTE: Leaving one side of a socket open is NOT exiting nicely. For example, if the server side if a socket dies, the client's side should not print the default exception to the screen (such as occurs in JAVA when exceptions are not handled) or just hang.

What to submit:
- Do not submit any executables
- The pcap file containing one TLS session between the client and server.
- client source code titled client.<ext> where <ext> depends on the language you used
- server source code titled server.<ext>
- The client and server certificates/keystores with which you tested your programs
- Any helper functions that are in separate files
- A makefile - mandatory for C, C++, optional for JAVA. If you use JAVA and no makefile, your code will be compiled by typing "javac *.java" If you include a makefile, your code will be compiled by typing "make"
- README file: (1) What you had to install on the VM, i.e. apt-get install <NAME> for everything you installed. (2) the steps you used to create the client and server certificates (3) how to run your programs. If you have files other than client and server, describe any helper functions that are in separate files - include a list of such files with one or two lines stating the purpose of each.

If you are using JAVA, you will need to use client and server as the class names (these are in lower case). If you are using python, use client.py and server.py as the names for the source code. If you are using C/C++, the executables produced by the makefile should be named client and server.