

CS 447 : Networks and Data Communications Programming Assignment #03

Total Points: 150



Assigned Date : Thursday, June 15, 2017

Due Date : Tuesday, June 27, 2017 @ 10:59:59 a.m. (Strong Deadline)

Overview

For your third (and final) project, you will implement a secure client/server application using TLS (Transport Layer Security). **Note:** This final assignment has a **firm deadline** and does not include the typical 48-hour late penalty period. Here's the back story.

Back Story

Haddock was quite happy with the new and improved email system until the Thomson and Thompson showed up one day with some devastating news. "You know Haddock, even with authentication, we can still read all your emails using Wireshark, right Thompson?", said Thomson. "Precisely Thomson. In fact, anyone can use Wireshark to read all your plaintext emails," replied Thompson.

Haddock ran down the hallway and stormed into Calculus's office. "Blue blistering barnacles! Calculus", screamed Haddock. "Those two Thompsons are using a shark to read my emails?". Calculus calmed Haddock and explained to him that he plans to use Transport Layer Security (TLS) in the next iteration of the email system to make it even more secure.

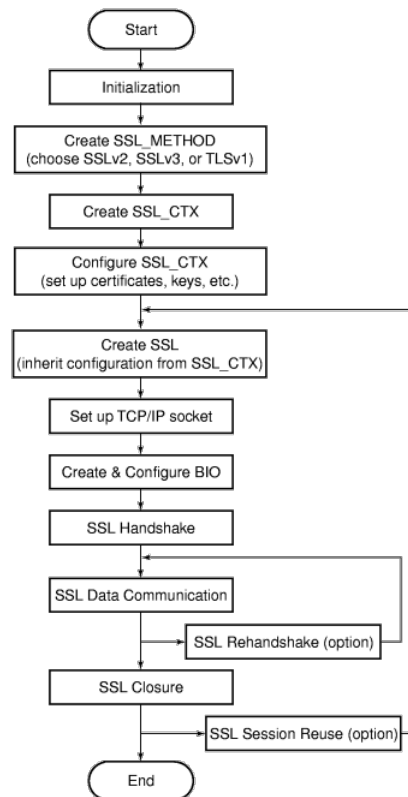
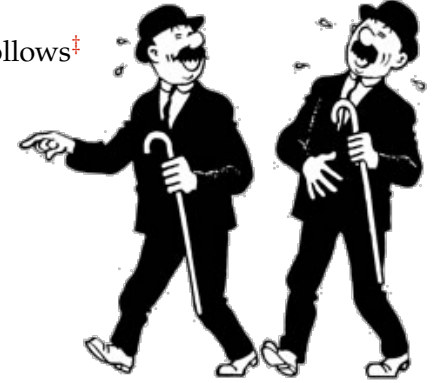
Technical Requirements

In order to keep this assignment within the time spec of the project and to accommodate the fact that TLS requires a reliable Transport Layer protocol, the technical requirements of this assignment are simplified in the following manner.

1. You are only required to develop a client-server application to mimic the sender → server SMTP interaction. However, this interaction is now occurring over TCP (instead of UDP).
2. You are not required to develop the server → receiver portion.
3. All PR02 technical requirements except for items 4 and 5 applies to PR03. This includes sender authentication, password strategy, and server incident management.
4. Before initiating the SMTP protocol interaction, your client and the server must establish a secure channel using TLS. See Logistics below for more information.
5. Your solution must use a TLSv1 (or newer depending on your language support) negotiation. Refrain from using any of the (old) SSL negotiations.

Logistics

- All applicable PR02 Logistics requirements applies to PR03. Ignore any logistical requirements related to HTTP.
- You are programming only one server and a client, i.e., SMTP over TCP.
- **Extra Credit (25 pts):** Run Wireshark on a secure SMTP interaction and on an unsecure SMTP interaction. Include your observations with appropriately annotated screenshots on your project report.
- The general workflow of SSL integrated socket communication is as follows[‡]



The nature of this assignment will force you to do significant amount of unsupervised learning – online research, forum scans, trial-and-error, and troubleshooting. Most likely, you will also discover (on your own) that there are more than one library package implementation of openssl standard, especially for C/C++, which might further complicate your task. So, don't get frustrated but instead use this as very valuable learning opportunity. Here are some resources that I believe will help you.

1. **HP SSL Programming Tutorial:**
http://h41379.www4.hp.com/doc/83final/ba554_90007/ch04s03.html
2. **Fedora Security Team – Defensive Coding**
https://docs.fedoraproject.org/en-US/Fedora_Security_Team/1/html/Defensive_Coding/index.html

[‡]Figure from the HP SSL Programming Tutorial.

3. **OpenSSL Wiki – Simple TLS Server:**
https://wiki.openssl.org/index.php/Simple_TLS_Server
4. **OWASP – Using the Java Secure Socket Extensions:**
https://www.owasp.org/index.php/Using_the_Java_Secure_Socket_Extensions
5. <http://simplestcodings.blogspot.com/2010/08/secure-server-client-using-openssl-in-c.html>
6. http://stilius.net/java/java_ssl.php
7. <https://media.readthedocs.org/pdf/pyopenssl/latest/pyopenssl.pdf>

Almost all of these helpful resources also include sample code that might aide you in your overall design and development. Using online resources does not mean you are allowed to copy and use someone else's code for your purpose. Such incidents, if detected, will be treated as academic dishonesty, so please do not copy-paste some else's code in your solution. Instead, try to grasp the core idea behind their solution and weave it in to your own work.

If you find a resource that you think will help your classmates, please share it with me so that I can disperse it among you colleagues.

Instructions

- **Start early!!**
- **Take backups of your code often!!**
- Follow a good coding standard. Use a Google style guide appropriate for your favorite programming language found here <https://google.github.io/styleguide/>, if you don't already follow one.
- The due date of this assignment is **Tuesday, June 27, 2017 @ 10:59:59 a.m.** A dropbox will be opened for submission through Moodle. This is a **firm deadline** as the final assignment does not include the typical 48-hour late penalty period.

Deliverables

A complete solution comprises of:

- A short report (max 5 pages) in **PDF** format of the design and implementation of your system. Your report should include the followings:
 - Introduction
 - Design choices and protocol/reply codes used.
 - The output of a sample run (including screenshots where applicable).
 - Summary and Issues encountered (if applicable).

Note: Your report will carry a significantly more weight in compared to the other two projects as it will be a reflection of your unsupervised learning efforts. Please make sure to document your efforts and to develop a good project report.

- A short **readme** file with compilation instructions. A **makefile** is mandatory if your solution involves running multiple compilation instructions. The only exception, most likely, would be Python.
- A compressed tarball of the directory containing your source code. **Do not** include executables, folders created by your programs, or your test emails in this tarball. To create a compressed tarball

of the directory `source`, use the following command: `tar -zcvf siue-id-pr2.tar.gz source/`.
e.g. `tar -zcvf tgamage-pr2.tar.gz PR02/`.

Caution: File formatting standards (PDF, readme, make, and tar.gz) as well as some of the logistics requirements are set forth to streamline the grading process. Submissions that take a unnecessarily long time grade due to not following the standards listed in this document will be subject to penalties.

Collaborating on ideas or answering questions is always encouraged. Most times, I find that you learn a lot from your peers. However, do not share/copy/duplicate code from others. If you use code found online, remember to cite their source in your report. Issues related to academic integrity and plagiarism have **ZERO** tolerance.