CS447 : Networks and Data Communications

Programming Assignment #03

Total Points: 150

Assigned Date : Thursday, June 17, 2021
Due Date : Tuesday, June 29, 2021 @ 10:59:59 a.m. (Hard Deadline)

Overview

For your third (and final) project, you will implement a secure client/server application using TLS (Transport Layer Security). Up to this point, you've already gained some experience with socket programming and how to implement protocols to RFC specifications in PR02. The goal now is to secure your application using TLS.
Note: This final assignment has a **hard deadline** and does not include the typical 48-hour late penalty period.

Technical Requirements

ALL PR02 technical requirements fully apply to the PR03. Additionally, following new requirements should be met.

1. Your solution must use a `TLS 1.2` (or newer depending on your language support) negotiation. Refrain from using any of the (old) SSL negotiations.

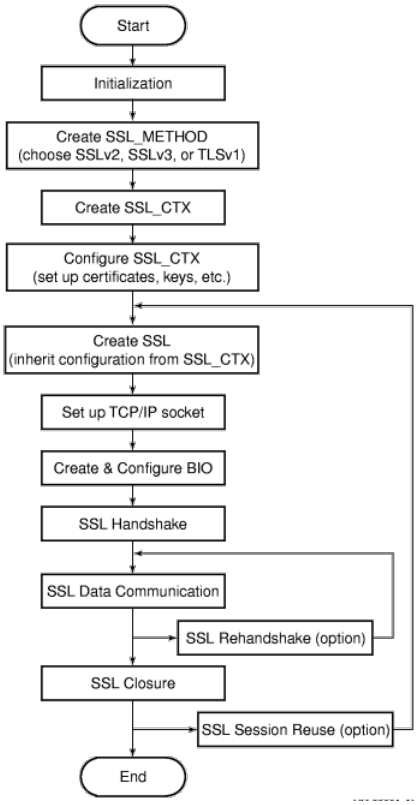
The nature of this last assignment will force you to do significant amount of unsupervised learning – online research, forum scans, trial-and-error, and troubleshooting. Most likely, you will also discover (on your own) that there are more than one library package implementation of `openssl` standard,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

Logistics

- You must provide proof of secure communication. To do this, run `Wireshark` with and without TLS separately, and provide appropriately annotated screenshots in your report. Failure to provide sufficient proof of secure communication will be considered as an indication of not meeting project requirements.
  - Two terminal equivalents for `wireshark` are `tcpdump` and `tshark`. You can save your capturing session as a `.pcap` file, which can be later opened in `wireshark` for easier analysis. The `wireshark` manual explains how to use both. See here https://www.wireshark.org/docs/wsug_html_chunked/AppToolstcpdump.html and here https://www.wireshark.org/docs/man-pages/tshark.html

- Submit only patch files. We like to see what you changed and how much you changed from your PR02 in order to meet PR03 specifications. Here's a short guide on how to create as well as apply patch files https://www.shellhacks.com/create-patch-diff-command-linux/.

The general workflow of SSL (outdated) integrated socket communication is as follows[‡]. Also listed are some resources that I believe will help you.



1. **HP SSL Programming Tutorial:**
   http://h30266.www3.hpe.com/odl/axpos/opsys/vmsos84/BA554_90007/ch04.html
2. **Fedora Security Team – Defensive Coding**
   https://docs.fedoraproject.org/en-US/Fedora_Security_Team/1/html/Defensive_Coding/index.html
3. **OpenSSL Wiki – Simple TLS Server:**
   https://wiki.openssl.org/index.php/Simple_TLS_Server
4. **Java Platform Security Developer's Guide (Ch. 8) Java Secure Socket Extension (JSSE) Reference Guide:**
   https://docs.oracle.com/en/java/javase/14/security/java-security-overview1.html
5. **PyOpenSSL Documentation:**
   https://buildmedia.readthedocs.org/media/pdf/pyopenssl/latest/pyopenssl.pdf
6. http://simplestcodings.blogspot.com/2010/08/secure-server-client-using-openssl-in-c.html

Almost all of these helpful resources also include sample code that might aide you in your overall design and development.

[‡]Figure from the HP SSL Programming Tutorial.

Using online resources does not mean you are allowed to copy and use someone else's code for your purposes. 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

- This is an individual assignment. **Do your own work**.
- **Start early!! Take backups of your code often!!**. Use of a version control software is highly recommended!
- Make sure to test your program properly on a linux platform before your final submission.
- You may use any programming language of your choice out of `C`, `C++`, `Java` or `Python` as these are all available on the home server. However, you **must** make sure that your code compiles and runs on a typical Linux machine. Absolutely **DO NOT** include executables with your submissions.
- A **Makefile** is mandatory. Whether or not your program needs to be compiled, have it echo instructions to run the program.
- Follow a good coding standard. Use one of Google's style guides for your language found here `http://google.github.io/styleguide/`, if you don't already follow one.
- The report part of your solution must be produced using a word processor. LaTeX is highly recommended but not a requirement.
- Your final report should be in **PDF** format. No exceptions.
- Any figures, graphs, plots, etc., should also be produced using appropriate computer applications. If using LaTeX, the pgfplots package is very useful for making all sorts of graphs.
- You are strongly encouraged to answer your own design and/or implementation questions based on the the RFCs, rather than defaulting to the instructor immediately. Make sure to properly document any such choices in your report and be as much descriptive as possible with proper sectional citation(s) from the relevant RFCs (*that way I will know why you did what you did*).
- The due date of this assignment is **Tuesday, June 29, 2021 @ 10:59:59 a.m. (Hard Deadline)** A dropbox will be opened for submission on Moodle. PR03 has a **firm deadline** and there will be no late submission period.

Deliverables

A complete solution comprises of:

- A short report of the design and implementation of your system. The report should be **`PDF`** format. At minimum, your report should include the following sections:
  - Introduction: Your objective and what you hope to gain from assignment.
  - Overall design, specific design choices, and reply codes used.
  - The output of a sample run (including screenshots where applicable). See Logistics requirement above.
  - Summary and Issues encountered. What you were able to achieve from your own objectives (from the introduction) as well as project specifications. Make sure to explicitly list functionality you failed to implement (or buggy).

- A compressed tarball that contains:
  - a directory containing (only) your patch files. **Do not** include executables, folders created by your programs, or any other files not specifically listed here as required.
  - A short `README` file with patching instructions and compilation instructions.
  - A `makefile` to automate compilation. For python-based submissions, echo your `README` file.

Caution: File formatting standards (`PDF`, `README`, `make`, and `tar.gz`) as well file naming requirements listed under technical requirements above 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 each others 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 including online sources. The exercise is meant for you to learn network programming, not to test your googling abilities. Issues related to academic integrity and plagiarism have **ZERO** tolerance.

Useful Resources

- Linux Man pages – found in all linux distributions
- Beej's Guide to Network Programming – A pretty thorough free online tutorial on basic network programming http://beej.us/guide/bgnet/output/print/bgnet_USLetter.pdf
- Internet Relay Chat: Client Protocol RFC #2812 https://tools.ietf.org/html/rfc2812
- Internet Relay Chat: Architecture RFC #2810 https://tools.ietf.org/html/rfc2810
- Internet Relay Chat: Channel Management RFC #2811 https://tools.ietf.org/html/rfc2811
- Internet Relay Chat: Server Protocol RFC #2813 https://tools.ietf.org/html/rfc2813
- The University of Chicago χ-Project http://chi.cs.uchicago.edu/chirc/irc.html
- The Transport Layer Security (TLS) Protocol Version 1.3 RFC #8446 https://tools.ietf.org/html/rfc8446
- Base64 encoding/decoding:
  - C/C++
    - glib – https://developer.gnome.org/glib/stable/glib-Base64-Encoding.html
    - openssl http://fm4dd.com/openssl/manual-crypto/BIO_f_base64.htm
    - GNU coreutils http://www.gnu.org/software/coreutils/coreutils.html
  - Java 14 – https://docs.oracle.com/en/java/javase/14/docs/api/java.base/java/util/Base64.html
  - Python 2 – https://docs.python.org/2.7/library/base64.html
  - Python 3 – https://docs.python.org/3/library/base64.html
  - Additional Resource – http://rosettacode.org/wiki/Base64_encode_data#C
- TLS Troubleshooting tips: https://maulwuff.de/research/ssl-debugging.html
- Creating Self-Signed Certificates: https://linuxize.com/post/creating-a-self-signed-ssl-certificate/. If you use any other resources, make sure to cite those in your report. 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.
- A Beginners Guide to Digital Certificates: http://www.steves-internet-guide.com/ssl-certificates-explained/