# WRITEUP for Web Proxy

Dongjing Wang

03/13/2024

## File List

- **README.md:** A brief explanation of the project, including how to compile and run the program, as well as the directory structure.
- **src/:**
  - **acl.cc:** For the main file that prohibits access to the relevant logic of this file part.
  - **acl.hh:** Header files for the implementation of logic related to prohibiting access to files.
  - **log.cc:** The main file for the implementation of some related logic of the log file.
  - **log.hh:** Placeholder for log.hh description.
  - **myproxy.cc:** A total file that integrates all other .cc and .hh files.
  - **ssl.cc:** Main document about SSL/TLS encryption algorithm (mainly HTTPS channel establishment).
  - **ssl.hh:** Header files about SSL/TLS encryption algorithms (mainly HTTPS channel establishment).
  - **testscript.sh:** Integrated automated testing for five tests of my project.
- **bin/:**
  - Directory for storing executable files generated during the build process. As per project guidelines, this directory is to be kept empty in the repository.
- **doc/:**
  - **WRITEUP.tex:** The LaTeX source file for detailed project documentation. It outlines the project's design, implementation details, instructions for use, test cases, and any known limitations.
  - **WRITEUP.pdf:** Compiled from WRITEUP.tex, provides a readable format of the project documentation for easy sharing and review.

# Test Cases Overview

This section outlines the test cases conducted to validate the functionality and robustness of the proxy server. Each test case aims to verify a specific aspect of the proxy server's behavior under different scenarios. The corresponding shell script 'testscript.sh' in the 'src/' folder automates these tests. For execution details, please refer to the tutorial in 'README.md'.

1. **Simple Test for the Proxy Server:** Validates basic proxy functionality, including correct forwarding of HTTP GET requests to www.google.com through the proxy server and ensuring the response status code is 200.

2. **Multithreading Test for the Proxy Server:** Tests the proxy server's ability to handle multiple concurrent connections, using both GET and HEAD requests for different URLs. This test assesses the server's multi-threading capabilities and its ability to manage multiple requests simultaneously without loss of functionality.

3. **Server Unable to Connect to Requested URL:** Verifies the proxy server's error handling when attempting to access an invalid URL. The expected response is a status code of 400, indicating a bad request.

4. **Testing for Expected 501 Not Implemented Response:** Examines the proxy server's response to unsupported HTTP methods, such as POST and PUT. The expected outcome is a 501 Not Implemented status code, confirming the proxy server correctly handles unsupported methods.

5. **ACL Test for the Proxy Server:**

    - Part 1: Ensures the proxy server can forward requests to allowed URLs, specifically verifying access to www.google.com is permitted and results in a 200 status code.
    - Part 2: Tests the Access Control List (ACL) feature by blocking www.google.com and verifying that attempts to access it through the proxy result in a 403 Forbidden status code.

# Design Overview

This section provides an overview of the architecture and operational mechanics of our HTTP proxy server, designed to handle HTTP GET and HEAD requests, with support for SSL/TLS encrypted connections and access control lists (ACLs) for URL filtering.

## Proxy Server Design:

The proxy server acts as an intermediary between clients and the web, processing incoming HTTP requests, applying ACLs, and then forwarding those requests

to the intended servers. It also handles responses from web servers and directs them back to the respective clients.

- **Connection Handling:** The server listens for incoming client connections on a specified port. Upon accepting a connection, it spawns a new thread to handle the communication, allowing for simultaneous processing of multiple client requests.
- **Request Parsing and Forwarding:** Parses incoming HTTP requests to extract the method, URL, and headers. Based on the URL, it decides whether to forward the request to the web server or block it according to the ACL.
- **SSL/TLS Support:** Utilizes OpenSSL to establish secure connections with web servers that require HTTPS. This ensures that data transmitted between the proxy and web servers is encrypted for privacy and security.
- **ACL Filtering:** Employs an access control list to filter requests to forbidden URLs. The ACL can be dynamically reloaded to allow changes without restarting the server.
- **Logging:** Implements detailed access logging, recording each request along with its outcome (e.g., forwarded, blocked) and relevant metadata like client IP, request line, and response size.

## Security and Reliability:

- **Thread Safety:** Uses mutexes to protect shared resources, ensuring thread-safe operations, especially important for the ACL and logging functionalities.
- **Signal Handling:** Includes a signal handler to catch and process signals for tasks such as gracefully shutting down the server or reloading the ACL file without service interruption.
- **Error Handling:** Incorporates comprehensive error handling throughout the server's operations, including network errors, SSL/TLS issues, and parsing errors, to maintain stability and provide informative feedback.

**Limitations and Future Work:** It seems that the requirements in the PDF for this task should be met, but this proxy server is still not perfect. In order to save time and complexity, the pipelining and chuck encoding parts were deliberately not considered.