CSE 156/L

Programming Project: C2S Proxy

Due: [ ]

The goal of this project is to develop a proxy HTTP server that can accept HTTP requests from clients and convert them to HTTPS requests to the web server. In addition, the HTTP request can be filtered based on an access control list. The requests can be generated by any Web client, such as the curl command or a web browser. The proxy converts plain text HTTP requests from a client to HTTPS requests and then returns the responses to the client. In summary, the proxy converts cleartext HTTP requests from clients to HTTPS requests to servers and vice versa for the responses.

Description

The proxy server needs to support secure communication of HTTP 1.1 with HTTPS. It needs to handle only the GET and HEAD requests. The proxy must be designed to support concurrent requests from multiple web clients. The client could be wget, curl or a browser.

The proxy must also have the capability to parse the headers of incoming HTTP requests and forward them to their ultimate destination web servers if allowed by the access control file. It must then forward any responses received from the a destination server back to the web client.

The access control file specifies the list of forbidden sites by either their domains or IP and is maintained in a file. The proxy server should reload the forbidden sites file when it receives a 'Control-C' signal. An example looks like:

```
www.bookface.com
www.youtube.com
www.fakenews.com
10.6.6.6
```

The proxy server is started with the following command with three arguments.
`./myproxy` *listen_port forbidden_sites_file_path access_log_file_path*

Requirements

- The proxy **must** handle requests concurrently.
- The proxy **must** handle receiving packets from one or more web clients.
- The proxy listens for requests on *listen_port*. It must always remain open and accept requests from web clients.
- When the request URL doesn't specify a port, the proxy must make the outgoing connection to the default port 443 for HTTPS. But if the URL does specify a port, the proxy must make the connection to that port with HTTPS.
- The proxy server must return an HTTP error response 403 (Forbidden URL) to the browser if the access is to a site in the forbidden-sites file.
- If the proxy server received an HTTP request method other than GET or HEAD, it should return an error code of 501 (Not implemented).
- The proxy server must also respond to any errors in the request (for example HTTP header fields missing or inconsistent) with the appropriate responses (for example, 400 Bad Request).
- Sending of 'Control-C' causes the system to send an INT signal (SIGINT) to the target running process. By default, this signal causes the process to immediately terminate. For the project, this signal must not terminate the proxy. Instead, it must cause the proxy to re-read the forbidden sites file and update what sites are to be blocked.

- **Logging:** The proxy server must maintain and write to access log file, called access.log, to track all requests received. It must print one line for each request with the following information:
  - Date and time the request was received
  - Requesting (client) host name or IP address
  - Request line from the client (method, URI, and HTTP version)
  - HTTP status code returned to the client (e.g., 200, 403 etc)
  - Length of the response to the client in bytes, including headers and content

  The log format has to be:

  date_format client_ip request_first_line http_status_code response_size_in_byte

  For example,

  ```
  2019-02-27T20:19:44.852Z 127.0.0.1 "GET http://ucsc.edu/ HTTP/1.1" 200 2326
  ```

- Persistent connections: Note that a browser will use persistent connections by default if it is using HTTP 1.1. In this case, either the client or the server may initiate the close. The proxy server must be able to deal with this by closing the connection on the other side when the TCP connection on one side closes. Don't leave dangling connections.
- You are free to output any needed debug messages on `stderr`.
- The documentation should describe the connection management and cleanup by the proxy.

Honor Code

All the code must be developed independently. All the work must be your own. You can re-use code that you have written in your own assignments in the class. Also, please do not cut-and-paste code blocks off of the Internet. Any violation of the above will result in getting a zero for the test.

What to submit?

All files must be submitted as a single compressed tar file (with `tar.gz` extension). The tar file name should start with your UCSC login id, e.g., **loginid.tar.gz**. The submission must follow the following steps.

1. A README file including your name, student ID, and a list of files in the submission with a brief description. It should be in the top directory.
2. Organize the files into sub-directories (`src`, `bin`, and `doc`)
3. The source files should be in the `src` sub-directory.
4. A `Makefile` that can be used to build the client program from the required source files. It should be in the top directory.
5. You should not include the compiled program. The `Makefile` will be used to generate the executable program. The make step should put the executable program in the `bin` sub-directory.
6. Documentation of your application in plain text or pdf. *The documentation should list 5 test cases done by you to validate the functionality.* Do not include any Microsoft Word files or other formats. The documentation should describe how to use your application and the internal design, as well as any shortcomings it might have. This file should be in the `doc` sub-directory.
7. Name your submission as above **loginid.tar.gz**, where loginid is your CruzID. Submit this file.

Grading

The assignment should compile and run on the campus linux timeshare (unix.lt.ucsc.edu). The grading will be done on that cluster. Each submission will be tested to make sure it works properly and can deal with errors. Grades are allocated using the guidelines below.

Basic Functionality:	60%
Dealing with Errors:	30%
Documentation:	10%

Note that 30% of the grade will be based on how well your code deals with errors. Good practice includes checking all system calls for errors and avoiding unsafe situations such as a buffer overflow.

The files must be submitted before the due date and time. Please make sure to submit on time. Late policy commences immediately after the due date/time. Reminder: Late policy deducts 10% per day from your grade.

The basic functionality is that the file created on the server is identical to the original file the client sent to the server. Pay attention to how well your code deals with errors. Good practice includes checking all system calls for errors and avoiding unsafe situations such as a buffer overflow.

The files must be submitted before the due date and time. Please make sure to submit on time.

FAQ

- The location of the access.log must be as given by the command.
- It's helpful to use wget or curl commands to download files from a remote site. An example command using curl as a client, where the proxy is running locally on port 9090. These commands can take an environment variable to use a proxy. Also, for a given URL, the optional argument can be used to go through a proxy. If the proxy is listening on the localhost port 9090:

  ```
  curl -x http://127.0.0.1:9090/ http://www.example.com
  ```

- If the client makes a CONNECT request, the proxy must send status code 501 (Not implemented) to it, as this method is not supported, and close the connection. For example, the proxy could get a request like the following from a client:

  ```
  CONNECT www.example.com:443 HTTP/1.1
  Host: www.example.com:443
  Proxy-Connection: Keep-Alive
  ```

- A standard browser client (e.g., Firefox) can make the HTTP requests to a proxy by configuring the browser to direct its requests to the proxy server. Usually this involves setting the IP address and listening port number of the proxy server in the browser.
- If the proxy is unable to resolve a domain name, it must return a "502 Bad Gateway" status to the client.
- If the proxy is has the IP address of the server but is unable to connect to it, it must return a "504 Gateway Timeout" status to the client.
- Consider if the client or server becomes unreachable in the middle of transferring a file. The impacted connections must be cleaned up.
- Look at strftime() and gmtime() for displaying Internet formatted timestamps, i.e., RFC 3339.

- The maximum number of concurrent requests for the purpose of testing could be up to 50.
- The maximum number of entries in the forbidden sites file is 1000.
- The logging should append to the access log file if it already exists.
- Upon reloading the new forbidden sites file, the existing connections to forbidden sites must be closed.

Tests to consider

- Client URL with http: scheme.
- Client URL with https: scheme.
- Unable to connect to the web server.
- The connections to the web client can break in the middle of the transfer.
- The forbidden files content changes and takes effect by the proxy after Control-C.