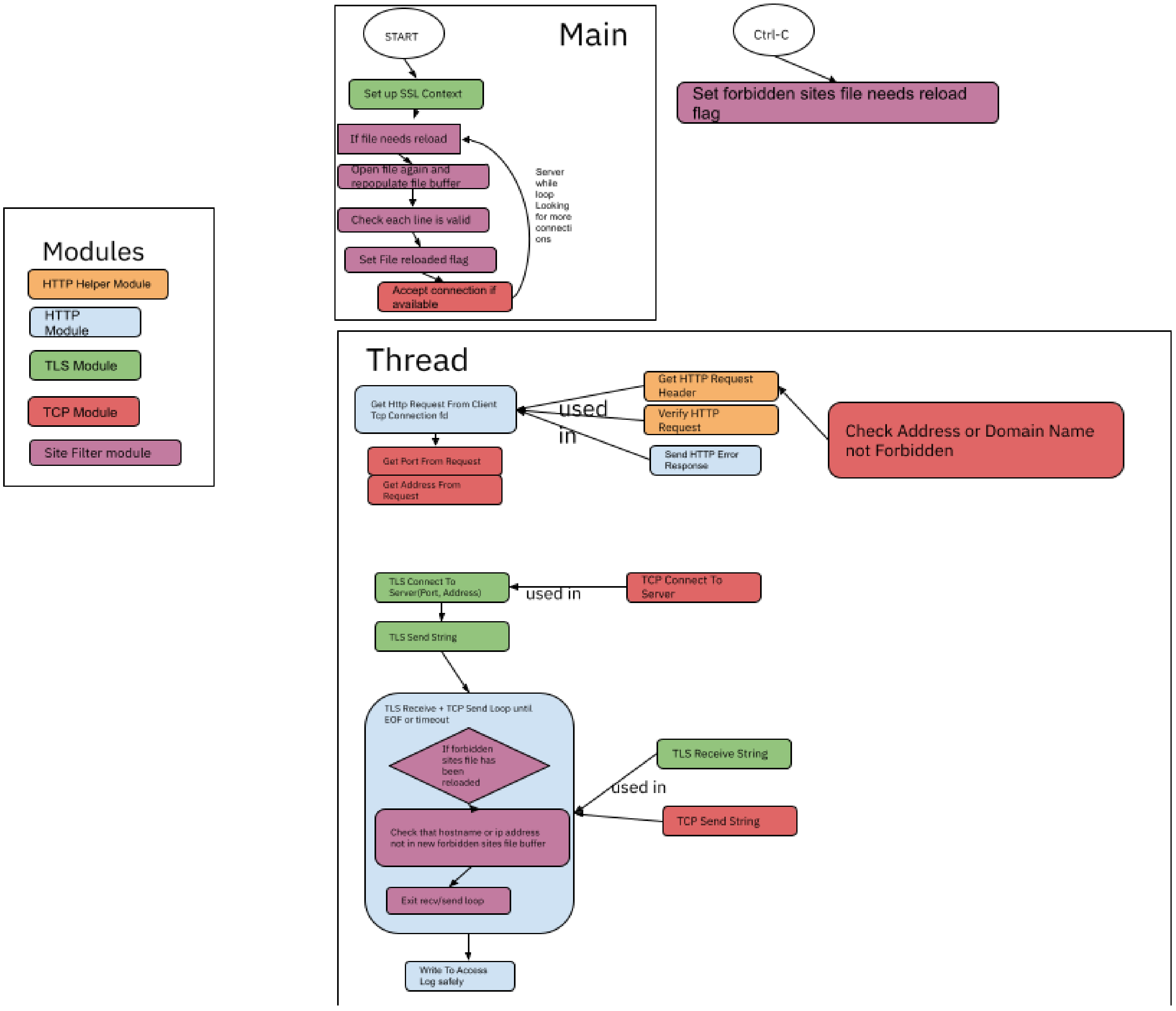
Prasana Guthy
Winter 2021

Final Project: TLS Proxy Design

Architecture

The below diagram describes the program flow and modules of my code.


Main
START
Set up SSL Context
If file needs reload
Open file again and repopulate file buffer
Check each line is valid
Set File reloaded flag
Accept connection if available
Server while loop Looking for more connections
Ctrl-C
Set forbidden sites file needs reload flag
Modules
HTTP Helper Module
HTTP Module
TLS Module
TCP Module
Site Filter module
Thread
Get Http Request From Client Tcp Connection fd
used in
Get HTTP Request Header
Verify HTTP Request
Send HTTP Error Response
Check Address or Domain Name not Forbidden
Get Port From Request
Get Address From Request
TLS Connect To Server(Port, Address)
used in
TCP Connect To Server
TLS Send String
TLS Receive + TCP Send Loop until EOF or timeout
If forbidden sites file has been reloaded
Check that hostname or ip address not in new forbidden sites file buffer
Exit recv/send loop
used in
TLS Receive String
TCP Send String
Write To Access Log safely

Test Cases

I used a few example files to test the proxy functionality.
A simple testing script was used in various ways:

```
#!/bin/bash
# Remove output file
#rm output.txt
#start test with netcat
#./bin/check_http & nc 127.0.0.1 9090 < input.txt > output.txt

#run test against curl
./bin/myproxy 9090 forbidden.txt log.txt &
curl -v -x http://127.0.0.1:9090 http://www.example.com -o output3.txt 2>> curlout.txt
curl -v -x http://127.0.0.1:9090 http://web.mit.edu -o output4.txt 2>> curlout.txt

#curl -v -x http://127.0.0.1:9090 http://web.mit.edu -o output4.txt &
#lldb ./bin/check_http
#./bin/check_http & curl -v -x http://127.0.0.1:9090 http://www.example.com -o
output3.txt
```

Log.txt is just an access log file and forbidden.txt was used to test the hot reload and site filtering functionality.
Forbidden.txt:

1. Case: Single (allowed site) curl - As always start the proxy in the background and call curl, redirecting the curl output to a file so that myproxy's error messages aren't mixed together with curl's output.
2. Case: 2 Curl commands: The proxy starts in the background and I send 2 http requests for allowed sites.
3. Case: 2 curl commands: 1 Allowed and one not allowed - I send 2 curl commands with one command requesting a forbidden site
4. Case: Reloading - I open the proxy and run a curl command for a forbidden file. After making sure that it returned a 403, I remove the website(in my case it was web.mit.edu) from the forbidden sites file and press ctrl-c (I can only press ctrl-c after foregrounding the proxy) . After the Reload, I run the same curl command and make sure it succeeds this time.

5. Case: multiple reloads - I try the same curl command multiple times while reloading more than once with ctrl-c, sometimes changing the file, other times not changing the file.

Implementation Limitations

- The HTTP request validation logic may not catch invalid headers (that is, headers that aren't usable by the web server, but are formatted as though they were custom HTTP headers)
- Making an HTTP request without specifying a Host: header will not work, because a hostname is required to enable SNI (Server Name Indication) which most modern web servers require for connection
- The hot reload is resource-intensive, because each thread needs to be able to stop an ongoing web server and client connection after a hot-reload.

Usage

Building the executable

The executable can be built using ‘make’ or ‘make all’ from the command line.

`make clean` will clean all object files and executables.

Running the proxy

The proxy can be started using the following:

`./bin/myproxy listen port forbidden sites file path access log file path`

It can be killed with the `kill` command.

Attribution

Cppreference.com - provided a lot useful documentation and code examples.
Man7.org - man pages for various functions
Linux.die.net - man pages for various functions
Stackoverflow.com - provided guidance on the nuances of using the pthread library.
beej.us/guide/bgnet/html - used as reference for initializing sockets safely and setting up and accessing sockaddr structs. Also used as reference for signal handling.
https://wiki.openssl.org/index.php/SSL/TLS_Client - OpenSSL official documentation example code used as basis for TLS code.
https://github.com/z4pu/tcp_to_tls/tree/master/src Used as reference for some TLS implementation details.
https://www.openssl.org/docs/man1.0.2/ Used for documentation reference.
https://www.delftstack.com/howto/c/sigint-in-c/ Used as reference material for signal handling
I also reused my own HTTP and TCP code from lab 1 for this class.