HTTP Proxy

Program Overview

This program consists of an HTTP proxy that implements queries using DNS. It receives HTTP requests generated by a Web Client and filters requests based on an access control blacklist. It supports HTTP 1.1 but only handles GET and Connect requests. The proxy uses threads to handle concurrent TCP connections with the clients and their corresponding server. The proxy is compatible with common web browsers and with curl and wget linux commands

Usage

To run the proxy, execute the following command:

```
./proxy <listen-port> <forbidden-sites-file>
```

The argument <forbidden-sites-file> must state the path to a file containing the website blacklist in the example format:

```
www.youtube.com

www.cnn.com
```

To not produce any unforseen errors please avoid using any other formatting for the blacklist. The argument <listen-port> needs to specify a usable port number in decimal notation.

Once the proxy is running, use a web broswer directed to the running proxy to access a website, or run one of the following commands:

```
curl -v --proxy <ip-address>:<listen-port> <wesbite-url>

wget -e use_proxy=yes -e https_proxy=<ip-address>:<listen-port> <wesbite-url>
```

You may use IP: 127.0.0.1 if running proxy in local host.
As the proxy runs, every succesful HTTP request will be logged in a log file access.log, which will keep log entries through multiple executions of the program.

Logic and Design

i. Connections

Upon receiving a TCP connection request, the proxy uses a new thread to open a new socket connected to the web client. The proxy then reads the HTTP request and parses the message. The proxy then checks if the request and the host are valid. Subsequently, the proxy performs a DNS lookup, setting up a connection if the lookup was succesful. The proxy then begins redirecting all traffic between hosts, keeping the connection alive until it receives a TCP disconnect signal (empty packet). Since the proxy starts redirecting traffic trivially once the connection is established, it supports HTTP 1.1 chunked transfer mode. A C++ global map is used to store the forbidden sites, as it is constructed with a balancing tree, allowing for rapid access for massive blacklists. The server can only handle 100 connections at a time, sending a connection refused signal upon TCP connection request. This can easily be changed in the listen() function.

ii. DNS lookup

Address lookups using DNS are performed in the tcp_connect() function specified in proxy.cpp. It uses the function getaddrinfo() with the serv argument set to port 80 (HTTP). This function allows for connecting

to websites using IPv4 or IPv6 addresses, only connecting through TCP. The implementation of tcp_connect() was taken from UNIX® Network Programming Volume 1, Third Edition, Pg 407, with only a few minor changes for HTTP connections and implementing a socket timeout of 2 minutes.

iii. Parsing of HTTP messages

Parsing is implemented in a simple and understandable way. Since HTTP messages are all ASCII characters and they have consistent formatting, we can use getline() to read lines from the input buffer, until we hit an empty line signaling the end of the header.

The library regex allows for the usage of regular expressions to split each line into words, where we can easily traverse through each word of the message. The paramaeters needed for handling the connections and requests are saved in a map for easy access. Both of these design decisions allow for efficient runtime and a simple implementation, yet they add a lot of memory size to the final executable binary as they include extensive C++ libraries. Here we see more cearly the difference of using C++ versus C; C++ has more versatility and convenient libraries that make the implementation much cleaner and efficient, but C is more trivial which uses less memory.

Errors

Most errors are handled without affecting the excecution of the pogram, unless these errors correspond to wrong usage or incorrect format of the blacklist file. Our own version of the function err_sys() allows for more clear STDERR messages. This function prints to STDERR the default system error defined by errno and a helpful error message. The program does have a chance to handle unforseen errors that might have gone unnoticed through debugging.

An override to the SIGINT signal allows for a safe interrupt, closing the socket and the log file properly before quitting the program.

If errors stop the connection with the client then an appropiate HTTP error message will be sent to the web client before closing the TCP connection. The following HTTP errors are implemented:

- 400 Bad Request
- 403 Forbidden Url
- 404 Not Found
- 405 Method Not Allowed
- 500 Internal Server Error *This one represents an unforseen error of the proxy

Comments

Since the g++ compiler uses arguments -std=c++17 and -lpthread, it might take a few seconds to compile depending on the system. In my case compiling didn’t take more than a second.

Acknowledgements

The unp.h header file and the function tcp_connect() were taken from UNIX® Network Programming Volume 1, Third Edition. Other code snippets and inspiration was taken from Professor Mike Parsa and his course CSE 156, Winter 2021 at UC Santa Cruz.