Ramzey Ghanaim

CMPE 156 Lab 4

March 12, 2018

Introduction

The purpose of this lab was to create a basic browser proxy that takes in GET and HEAD HTTP requests from a browser, forwards the message to the correct destination and sends back requested data to the browser.

Design and Error Handling.

When the proxy first starts up, It firsts check the arguments to ensure the user entered a port number and if successful. Next, the forbidden sites text file is read, followed by the program continues to make a connection with the browser with the provided port number. At this point the program waits for clients at the accept() UNIX function.

Once a client is accepted I fork a child process and receive the first request from the browser. I then get the URL if the request is a GET or HEAD request. If the request is not a GET or HEAD request, error 501 is sent to the browser. Once the URL is extracted, it is checked with the forbidden list to see if the URL is blacklisted. If it is, I send error 403 (forbidden) to the browser. Anytime an error is sent, I close the connection with the browser and exit the forked process.

If there are no errors at this point, the forward header is appended to the header received from the browser. Once this completes, the message is sent to the destination. First a DNS look up is done with the gethostbyname() function to find the IP address of the destination. Next the message is sent to the destination server followed by a receive to get information back from the server and forward it to the browser. The protocol for receiving data from a server and sending it to the browser will be discussed in the protocol section. After sending all the data back to the browser, the socket is closed on the server side. Next The request is logged into a log file with the request from the browser and the status code that was returned from the server. Date and time as well as the number of bytes of data being sent is also logged in each entry. After all data has been sent to the browser, the browser side connection is closed. Lastly, the forked process is exited, and the main process continues to wait for more requests.

Message Protocol

The main protocol that is being used is the protocol that is used for detecting when a connection is closed. I continue to receive data from a website server, while immediately sending what was received to the browser. When a time out occurs, I assume the data is done downloading and I close the connection with the server. I chose this protocol because my headers from the server never contained a “Connection: close” line.

Usage

These instructions assume the user is in the main directory of this project

1. To build the program simply type:

   “make”

2. To run the server, in one terminal in the `bin/` folder type:

   “proxy <portNum> <forbidden-sites-file>”

Note: The <portNum> must match ports specified in Browser’s proxy settings.

Note 2: forbiddensites.txt is a provided file I used with testing.

3. In a browser with a proxy configured type the following in the URL bar:

   “www.example.com”

   The website will be displayed.

4. When one wants to clean the projects simply type:

   “make clean”