CSE156/L Final Project: C2S Proxy

Serjo Barron
sepbarro
Winter 2024

Program Design - Top Level (Proxy Server):

```
1. Parse arguments: port number, forbidden sites file, access log file
2. Initiate SIGINT handler
3. Check if the port is valid
4. Create a server socket and bind it to the port number
5. Run an infinite loop that does the following:
    a. Accept client socket upon detected request
    b. Create a thread that handles the client
    c. handle_client() does the following:
        i. Receive client request
        ii. Parse HTTP request to retrieve hostname, IP, and URI
        iii. If parsing HTTP fails (missing fields):
            1. If the method is not GET or HEAD:
                a. Send 501 “Not Implemented” response to client
                b. Log 501 request
                c. Close client connection
            2. Otherwise:
                a. Send “400 Bad Request” response to client
                b. Log 400 request
                c. Close client connection
        iv. If the hostname or IP is forbidden:
            1. Send “403 Forbidden” response to client
            2. Log 403 request
            3. Close client connection
        v. Create destination socket and bind to port number
        vi. Resolve hostname domain resolution
        vii. Connect to the destination socket
        viii. Send a request to the destination server
        ix. Receive a response from the destination server
        x. Send “200 OK” response to client
        xi. Log 200 request
        xii. Close client socket
        xiii. Close destination socket
    d. Once the thread finishes detach it
6. Close server socket
```

Forbidden Sites Loading:

```
1. Open forbidden sites file in reading mode
2. Create new forbidden sites array
```

3. Count the number of lines in the file using fgets
4. Reset the pointer to the beginning of the file
5. Allocate memory for the new array (based on the number of lines counted)
6. Read each line and store it in an array
7. Close forbidden sites file
8. Free memory of the previous forbidden sites array
9. Update/set the forbidden sites array to the new one

Forbidden Sites Detection:

1. Use the forbidden sites array from loading the file
2. Iterate over the number of detected forbidden sites
    a. Check if there is an instance of the hostname or IP in a given forbidden sites array index
    b. If there is an instance:
        i. Return 1 indicating a forbidden file was detected
    c. Otherwise:
        i. Return 0 indicating no forbidden files were detected

HTTP request parsing:

1. Parse the request into its method, URL, and version
2. Check that the method is GET or HEAD, if not then return the exit function on failure
3. Find the position of “//” then calculate the distance “//” and the first instance of “/” to find the hostname
4. Retrieve port (after “:”) if present, otherwise set to 80 by default

Program Instructions:

1. Go to the top directory (where the Makefile is) and use the command “make” to create a binary executable (for both server and client) in the bin directory
2. Change the directory to the bin directory
3. Open two terminal windows, you will need one to start the proxy server and one to send client requests
4. To run the proxy server:
    a. ./myproxy <port number> <forbidden sites file> <access log file>
    b. The port number cannot be well known: 0-1024, but it may be within the range of 1024-65535, anything outside of that range is not permitted
    c. The forbidden sites file and access log file/path must exist prior to starting the server, they may be empty if desired
5. Sending client requests:
    a. You can use command line tools such as curl to send GET or HEAD requests
        i. curl -x http://127.0.0.1:9090/ http://www.example.com

Test Cases:

1. Invalid input: argument count
    a. ./myproxy 8080 forbidden
        i. Missing access log file/path (4 expected, 3 given)
2. Invalid input: negative port number
    a. ./myproxy -8080 forbidden logfile
        i. The proxy server socket cannot be bound to a negative port number
3. Invalid input: well-known port number
    a. ./myproxy 80 forbidden logfile
        i. Cannot bind to a well-known port number, these are reserved
4. Invalid input: out-of-range port number
    a. ./myproxy 65536 forbidden logfile
        i. The port number is out of range of dynamic and ephemeral ports (1024-65535)
5. Invalid input: PUT request
    a. ./myproxy 8080 forbidden logfile
    b. curl -X PUT http://127.0.0.1:8080/
        i. The forbidden file is empty
        ii. A PUT request is not supported by the proxy server so a 501 Not Implemented response is the result
6. Invalid input: Forbidden domain
    a. ./myproxy 8080 forbidden logfile
    b. curl -x http://127.0.0.1:8080/ www.amazon.com
        i. The forbidden file contains:
            1. www.amazon.com
        ii. The proxy server will stop prematurely after detecting that the domain matches up with something within the forbidden sites and result in a 403 Forbidden response
7. Invalid input: Unresolved domain name
    a. ./myproxy 8080 forbidden logfile
    b. curl -x http://127.0.0.1:8080/ www.amaz
        i. The forbidden file is empty
        ii. The proxy server will attempt domain name resolution and fail resulting in a 502 Bad Gateway response
8. Valid input: GET request
    a. ./myproxy 8080 forbidden logfile
    b. curl -x http://127.0.0.1:8080/ www.amazon.com
        i. The forbidden file is empty
        ii. A GET request is sent via HTTP to the proxy server
9. Valid input: HEAD request
    a. ./myproxy 8080 forbidden logfile
    b. curl -x http://127.0.0.1:8080/ -I www.amazon.com
        i. The forbidden file is empty
        ii. A HEAD request is sent via HTTP to the proxy server