

Final Project - Proxy Server

Code Design and Functionality

1. Server Initialization

At startup, the server initializes its configuration by loading settings, such as the port it will listen on, paths to any relevant files (like a forbidden sites list), and other operational parameters. This phase is crucial for setting up the environment in which the proxy operates.

2. Request Handling

The core functionality revolves around handling incoming client requests. This involves several steps:

- **Listening:** The server listens on a specified port for incoming connections from clients.
- **Connection Handling:** Upon accepting a connection, the server reads the HTTP request from the client, parsing necessary information such as the request type (GET, POST, etc.), the target URL, and any headers.
- **Content Filtering:** If the request URL matches any entry in the forbidden sites list, the server blocks the request and sends an appropriate response back to the client, such as an HTTP 403 Forbidden status.

3. Forwarding Requests

For allowed requests, the server then forwards the request to the target web server on the internet. This might involve:

- **DNS Lookup:** Translating the hostname of the target URL into an IP address if necessary.
- **Connecting and Forwarding:** Establishing a connection to the target server, sending the request, and waiting for the response.

4. Response Handling and Caching

Once the proxy server receives the response from the target server, it processes it:

- **Caching (Optional):** If the response is cacheable, the proxy might store it for future requests.
- **Response Delivery:** The server then sends the response back to the original client.

5. Logging and Monitoring

The server keeps logs of requests and responses, including details like request URLs, timestamps, and possibly the size of the data transferred. This is important for monitoring, debugging, and analyzing the traffic going through the proxy.

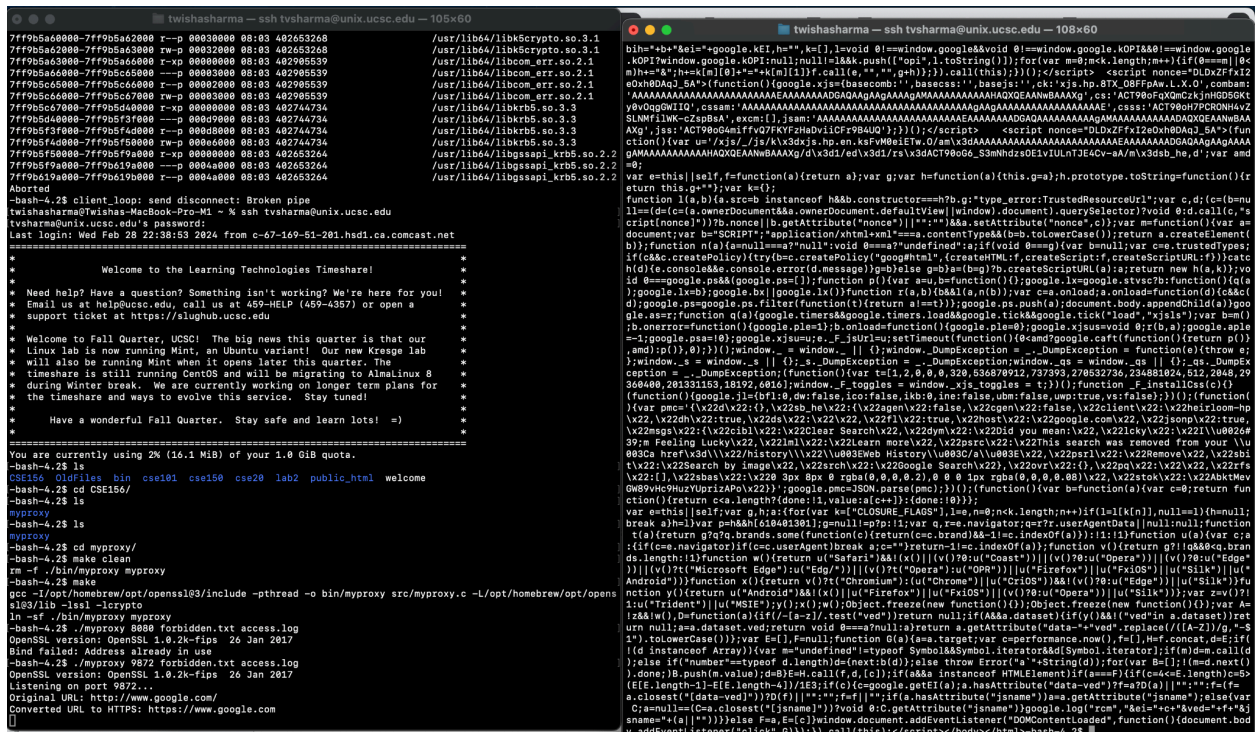
6. Concurrency Handling

To efficiently serve multiple clients simultaneously, the server implements concurrency, often through threading or asynchronous I/O. This ensures that the proxy can handle numerous requests in parallel without blocking.

Test Cases

Case 0: Basic GET Request

- Objective: Verify that the proxy server correctly handles a basic HTTP GET Request and relays the content from the target server
- Procedure: `./myproxy 9872 forbidden.txt access.log curl -x http://127.0.0.1:9872 http://www.google.com`
- Outcome:



The screenshot shows a terminal window on the left and a web browser on the right. The terminal window displays the output of the `./myproxy` command, which starts the proxy server on port 9872. The browser window shows the Google homepage with search results for "twishasharma". The terminal output includes the following lines:

```
twishasharma@twishas-MacBook-Pro-M1: ~ % ./myproxy 9872 forbidden.txt access.log
Last login: Wed Feb 28 22:38:53 2024 from c-67-169-51-201.hsdl.ca.comcast.net

=====
Welcome to the Learning Technologies Timeshare!
=====
* Need help? Have a question? Something isn't working? We're here for you!
* Email us at help@ucsc.edu, call us at 452-HELP (452-4357) or open a
* support ticket at https://slughub.ucsc.edu
=====
Welcome to Fall Quarter, UCSC! The big news this quarter is that our
Linux lab is now running Mint, an Ubuntu variant! Our new Kresge lab
will also be running Mint when it opens later this quarter. The
timeshare is still running CentOS and will be migrating to AlmaLinux 8
during Winter break. We are currently working on longer term plans for
the timeshare and ways to evolve this service. Stay tuned!
=====
Have a wonderful Fall Quarter. Stay safe and learn lots! =)
=====

You are currently using 2M (16.1 MiB) of your 1.0 GiB quota.

-bash-4.25 ls
CSC150 OldFiles bin cse101 cse150 cse20 lab2 public_html welcome
-bash-4.25 cd CSE156/
-bash-4.25 ls
myproxy
-bash-4.25 ls
myproxy
-bash-4.25 cd myproxy/
-bash-4.25 make clean
rm -f ./bin/myproxy myproxy
-bash-4.25 make
gcc -I/opt/homebrew/opt/openssl@3/include -pthread -o bin/myproxy src/myproxy.c -L/opt/homebrew/opt/openssl@3/lib -lssl -lcrypto
ln -sf ./bin/myproxy myproxy
-bash-4.25 ./myproxy 9872 forbidden.txt access.log
OpenSSL version: OpenSSL 1.0.2k-fips 26 Jan 2017
Bind failed: Address already in use
-bash-4.25 ./myproxy 9872 forbidden.txt access.log
OpenSSL version: OpenSSL 1.0.2k-fips 26 Jan 2017
Listening on port 9872...
Original URL: http://www.google.com/
Converted URL to HTTPS: https://www.google.com/
```

Case 1: Forbidden Website Access

- Objective: Ensure that the proxy server correctly blocks access to the websites listed in the forbidden sites file.

- Procedure : ./myproxy 9872 forbidden.txt access.log curl -x 127.0.0.1:8080 -I http://www.ucsc.edu
- Outcome:

The image shows two terminal windows. The left window, titled 'twishasharma - ssh tvsharma@unix.ucsc.edu - 105x60', shows the command './myproxy 9872 forbidden.txt access.log' being executed, which starts an OpenSSL proxy server listening on port 9872. The right window, titled 'twishasharma - ssh tvsharma@unix.ucsc.edu - 108x60', shows the command 'curl -x 127.0.0.1:9872 -I http://www.ucsc.edu' being executed, which returns 'HTTP/1.1 403 Forbidden'.

Case 2: Concurrent GET Requests

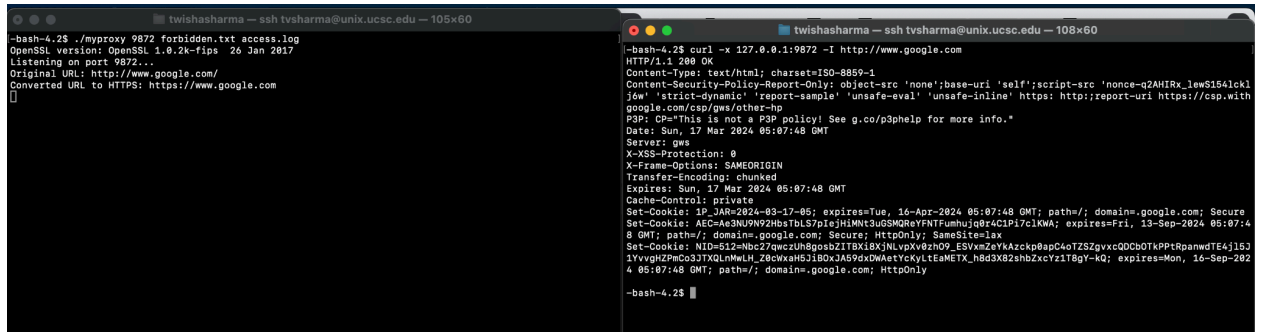
- Objective: Test the server's ability to handle multiple simultaneous GET requests
- Procedure: ./myproxy 9872 forbidden.txt access.log curl -x http://127.0.0.1:8080 http://www.google.com & curl -x http://127.0.0.1:9872 http://www.google.com & curl -x http://127.0.0.1:8080 http://www.google.com
- Outcome:

The image shows two terminal windows. The left window, titled 'twishasharma - ssh tvsharma@unix.ucsc.edu - 105x60', shows the command './myproxy 9872 forbidden.txt access.log' being executed, which starts an OpenSSL proxy server listening on port 9872. The right window, titled 'twishasharma - ssh tvsharma@unix.ucsc.edu - 108x60', shows the command 'curl -x http://127.0.0.1:8080 http://www.google.com & curl -x http://127.0.0.1:9872 http://www.google.com & curl -x http://127.0.0.1:8080 http://www.google.com' being executed. The output shows the proxy server handling multiple concurrent requests to google.com.

Case 3: Head Request

- Objective: Verify that the proxy server supports HEAD requests and correctly relays the headers from the target server.
- Procedure: ./myproxy 9872 forbidden.txt access.log curl -x 127.0.0.1:9872 -I http://www.google.com

- Outcome:



The image shows two terminal windows side-by-side. The left window is titled 'twishasharma — ssh tvsharma@unix.ucsc.edu — 105x60'. It shows the execution of a script named 'myproxy' which sets up a proxy on port 9872. The script output includes 'OpenSSL version: OpenSSL 1.0.2k-fips 26 Jan 2017', 'Listening on port 9872...', 'Original URL: http://www.google.com/', and 'Converted URL to HTTPS: https://www.google.com'. The right window is titled 'twishasharma — ssh tvsharma@unix.ucsc.edu — 108x60'. It shows the execution of a curl command: 'curl -x 127.0.0.1:9872 -I http://www.google.com'. The output shows an HTTP 200 OK response with various headers including 'Content-Type: text/html; charset=ISO-8859-1', 'Content-Security-Policy-Report-Only', 'Server: gws', 'X-XSS-Protection: 0', 'X-Frame-Options: SAMEORIGIN', and several cookies from google.com.

```
-bash-4.2$ ./myproxy 9872 forbidden.txt access.log
OpenSSL version: OpenSSL 1.0.2k-fips 26 Jan 2017
Listening on port 9872...
Original URL: http://www.google.com/
Converted URL to HTTPS: https://www.google.com

-bash-4.2$ curl -x 127.0.0.1:9872 -I http://www.google.com
HTTP/1.1 200 OK
Content-Type: text/html; charset=ISO-8859-1
Content-Security-Policy-Report-Only: object-src 'none';base-uri 'self';script-src 'nonce-q2AHIRx_1ewS154lck1j6w' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline' https: http:report-uri https://csp.withgoogle.com/csp/gws/other-hp
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Date: Sun, 17 Mar 2024 05:07:48 GMT
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
Expires: Sun, 17 Mar 2024 05:07:48 GMT
Cache-Control: private
Set-Cookie: 1P_JAR=2024-03-17-05; expires=Tue, 16-Apr-2024 05:07:48 GMT; path=/; domain=.google.com; Secure
Set-Cookie: AEC=Ae3NU9N2Hb8TbLS7piejHlMNT3uGSMQReYfNTFumhuja0r4C1P17c1KMA; expires=Fri, 13-Sep-2024 05:07:48 GMT; path=/; domain=.google.com; Secure; HttpOnly; SameSite=lax
Set-Cookie: NID=632=Nbc2ZqwezU8gpb8ZITBx18XjNLpXv0zh0Y_E3VwZeYkAtckpBapC4oT2SIZgvxcODC80TAPPRpamedTE4j1531VvghZPnCo3JTXQlnMwLH_Z9cWxaH5j180xJAS9dxDAeYkyltEaMETX_hBd3X82shbZxcYz1T8gY-KQ; expires=Mon, 16-Sep-2024 05:07:48 GMT; path=/; domain=.google.com; HttpOnly

-bash-4.2$
```