COMP 4621 Project Report

Kong Ming Hin (20510403) mhkongaa@connect.ust.hk

Multi-thread

- Whenever the proxy receives a request, creates a thread to handle that request.
- In each thread, the handler determines whether the request is HTTP or HTTPS.
- The thread subsequently creates 2 extra threads for handling forward and backward data transmission respectively (Refer to HTTP Requests Forwarding and HTTPS Requests Forwarding for details).

HTTP Requests Forwarding

0. The handler first check whether the request starts with a "GET" header before getting into this part.
1. Modify the request and change the absolute URL inside the header to relative URL. Retrieve host information and absolute URL in the meantime.
2. Cache the request in a file as history.
3. Return "HTTP/1.1 404 Not Found\r\n\r\n" back to user and terminate the thread if the host requested is found in the pre-defined block list.
4. Return the cache back to user and terminate the thread if the absolute URL requested is found in the cache.
5. Connect to host.
6. Create two threads. One for handling user-to-host transmission, which forwards everything from user to host. Another one for handling host-to-user transmission, which forwards HTTP response from host to user and cache the response to a file simultaneously. The data read after the HTTP header in the HTTP response is truncated to match the length specified (if any) in the "Content-Length" header.
7. Wait for the two threads to terminate.
8. Close the file descriptors of user and remote.

HTTPS Requests Forwarding

0. The handler first check whether the request starts with a "CONNECT" header before getting into this part.
1. Retrieve host and port information from the request.
2. Cache the request in a file as history
3. Return "HTTP/1.1 404 Not Found\r\n\r\n" back to user and terminate the thread if the host requested is found in the pre-defined block list.

4. Connect to host with the specified port.
5. Send a "HTTP/1.1 200 Connection Established\r\n\r\n" back to user.
6. Create two threads. One for handling user-to-host tunnelling, which forwards everything from user to host. Another one for handling host-to-user tunnelling, which forwards everything from host to user.
7. Wait for two threads to terminate.
8. Close the file descriptors of user and remote.

Access Control

0. Before running the proxy server, the owner should provide a list of blocked hosts, stored in "Block List.txt", in the same directory as the proxy server. Assume the file originates from Windows, so multiple hosts are separated with CRLF.
1. Right after the main thread creates a thread to handle the user's request, store the contents of the file into a string array.
2. The blocked hosts are compared with the host requested by user before sending the user's HTTP request to host or before connecting to the requested host, depends on whether the request is HTTP or HTTPS.

Caching

0. This process is done within the thread that handles host-to-server HTTP response transmission.
1. Right after the main thread creates a thread to handle the user's request, store the contents of the file into a string array.
2. The cached absolute URLs are compared with the absolute URL requested by the user before the user's HTTP request is sent to the host.
3. Open a file to store the requested absolute URL and connect to host if the absolute URL requested is not found in cache.
4. Inside the thread that handles host-to-user data transmission, open a file to store all incoming data from the HTTP request of the absolute URL. Each HTTP response of its corresponding HTTP request are stored in separate files, differentiate by the absolute URL in the HTTP request. The content cached in the files are sent back to user simultaneously.