**THỰC HÀNH LẬP TRÌNH HỆ THỐNG**

MSSV : 13520213

TÊN : Nguyễn văn Đức

**BÁO CÁO LAB 6**

1. Phân tích cách thức hoạt động của một proxy đơn giản:

- **Chức năng**: Proxy Server là một server đóng vai trò cài đặt proxy làm trung gian giữa người dùng trạm và Internet. Các kết nối giữa người gửi và người nhận sẽ bị chặn lại bới Proxy server, không được thực hiện một cách trực tiếp. Và thông tin dữ liệu khi đến được tiếp nhận ở môt cổng sau đó được chuyển tiếp đến phần còn lại của mạng thông qua một cổng khác với cổng đi vào. Nhờ cách truyền tin gián tiếp như vậy nên máy chủ proxy sẽ tạo nhiều khó khăn hơn nhiều cho tin tặc để có được địa chỉ nội bộ và chi tiết của một mạng riêng.

- **Phân tích mã nguồn**:

* Các hàm và luồng như sau

|  |
| --- |
|  |

* **Main():** Mở port + vòng lặp while vô tận để chấp nhận các kết nối từ client.
* **Rio\_readinitb:** Đọc dữ liệu
* **open\_port():** Khởi tạo socket để giao tiếp với client + lắng nghe (bind)
  + *input:* port của proxy
  + *output: socket*
* **New\_thread():** Mỗi request tới sẽ có 1 thread xử lý
  + *Input:* thông tin client
  + *Output: NULL*
* **proxy():** Chuyển tiếp request đến web server và response đến client.
  + *Input*: client
  + *Output*: Dựa vào việc forward đến webserver không thành công sẽ return -1
* **Forward\_to\_server():** Phân tích request của client + khởi tạo request và gửi request này đến server.
  + *Input*: socket
  + *Output*: in ra method, version, resource, URL, host, port của request
* **Forward\_to\_client():** Đọc dữ liệu từ response của webserver
  + *Input*: client, fb\_server
  + *Output*: return 0
* **open\_server():** Connect tới server và kiểm tra connect có lỗi hay không
  + *input:* host, port
  + *output: socket*
* **parse\_request\_line():** Hiển thị các thông tin của request
  + *input:* buffer, method, url, resource của request
  + *output:* Xuất các thông tin ra.

1. Phân tích hoạt động proxy lưu cache dữ liệu:

- **Chức năng**: Các phản hồi từ server sẽ được proxy lưu lại để phục vụ cho các request trong tương lai của client. Như vậy nội dung sẽ được proxy response về mà không cần phải request lên web server.

- **Phân tích mã nguồn**:

* Các hàm và luồng như sau:

|  |
| --- |
|  |

* **Tương tư như các hàm trên ở bài 1, nhưng ở đây ta có thêm chức năng cache. Như vậy mỗi chuyển tiếp request đến web server(hàm proxy), ta sẽ kiểm tra nội dung trang web đã được cache trên proxy chưa(parse\_request -> cache\_search ) rồi phản hổi về cho client(cache\_to\_client). Ngược lại thì sẽ forward request lên web server và nhận response trả về client đồng thời cache lại.**
* **proxy():** Chuyển tiếp request đến web server và response đến client.
  + **Nếu res = 0:** Tức nội dung web đã cache **-**> trả về cho client(**cache\_to\_client()**).
  + **Nếu res = 1:** Không có trong cache -> chuyển tiếp request đến web server(**forward\_to\_server()**).
  + *Input*: client
  + *Output:* Dựa vào biến res. (-1 request lỗi)
* **parse\_request():** Phân tích request của client. Kiểm tra nội dung đã có trong cache chưa(**cache\_search()**). Nếu chưa thì khởi tạo request.
  + *Input*: socket, host, port, fd\_req, resource
  + *Output*: return **-1** (request lôi) | **0** (có trong cache) | **1** (không có trong cache).
* **Cache\_to\_client():** In dữ liệu từ response lấy từ cache cho client.
  + *Input*: client, response
  + *Output:* return 0.
* **forward\_to\_server():** Chuyển tiếp request đến web server và response đến client.
  + *Input*: host, port, fd\_req
  + *Output:* return -1 (không thể connect tới host).
* **forward\_to\_client():** Đọc và cache dữ liệu từ response của webserver
  + *Input*: client, fd\_server, host, port, resource
  + *Output:* return 0.