Đại học Khoa học Tự nhiên, ĐHQG-HCM
Khoa Công nghệ Thông tin
Bộ môn Mạng máy tính

Báo cáo Đồ án 01 - Proxy Server

1. Phân công công việc

Công việc của nhóm được phân công dựa vào số lượng code phải viết hoặc/và khối lượng các công việc khác phải làm. Ví dụ như những người code ít sẽ được giao cho làm thêm những công việc khác như bắt gói tin, viết báo cáo. Những người code nhiều thì sẽ không cần làm thêm các công việc đó.

Hàm của người nào viết thì người đó có nhiệm vụ debug luôn hàm đó nếu tester phát hiện lỗi và báo cáo lại.

STT	MSSV	Họ tên	Công việc
1	19120729	Bùi Ngọc Thảo Vy	Viết các hàm chính `handle_http_request`, `parse_header`, `send_403_forbidden`.
2	19120626	Hồ Minh Quân	Viết các hàm `is_blocked`, `get_target_info`, `recvall`. Bắt gói tin và viết báo cáo cho phần này (mục 6 và 7 trong file này).
3	19120615	Hùng Ngọc Phát	Viết các hàm `main`, `log`. Test, báo cáo lỗi. Tổng hợp, chỉnh sửa và viết báo cáo chung (file này).

2. Thông tin về chương trình

- Chương trình được viết bằng ngôn ngữ lập trình Python 3 (**thư viện bổ trợ** phải cài từ pip: `termcolor`).
- Đa số các thành viên của nhóm đều sử dụng Linux, nên proxy server này được test và debug trên nền tảng Linux. Proxy server có thể sẽ hoạt động tốt trên các hệ điều hành tương tự Unix như Linux và macOS. Một số tính năng có thể không hoạt động trên Windows (như màu chữ trong khi log). Thầy/cô vui lòng xem kỹ hơn ở phần **Chạy chương trình**.
- Proxy server chỉ hỗ trợ HTTP (cụ thể là 2 phương thức `GET` và `POST`), không hỗ trợ HTTPS vì điều này gần như bất khả thi với low-level socket programming.
- Chương trình này gồm có các file sau:
  - `proxy.py`: chứa hàm main.
  - `requesthndl_module.py`: chứa các hàm để handle request từ client và các hàm phụ trợ liên quan.
  - `logging_module.py`: chứa các hàm phục vụ mục đích log lại thông tin.
  - `blacklist.conf`: danh sách các trang web bị chặn.
  - `403forbidden.png`: một file hình ảnh để làm thêm sinh động trang HTML trả về lỗi `403 Forbidden`.

- Nhóm chúng em phối hợp làm việc thông qua Github: https://github.com/hungngocphat01/proxy-server

3. Các hàm chính

3.1. Hàm main

- Hàm main là hàm chính, không nhận vào tham số, không trả về dữ liệu. Hàm này có nhiệm vụ:
  - Khởi tạo một socket để chấp nhận kết nối từ client.
  - Chấp nhận kết nối từ client.
  - Mở một thread mới để xử lí request từ client.
  - Khi có lỗi (exception) xảy ra ở trong nó (không phải các hàm mà nó gọi) thì in ra chi tiết và đóng tất cả các kết nối, ngừng chương trình.

3.2. Các hàm phụ trợ

```
def handle_http_request(client_socket: socket.socket, a: tuple) → None
```

- **Nhiệm vụ:**
  - Nhận request của client gửi lên.
  - Kiểm tra xem trang web mà client yêu cầu có bị chặn hay không.
  - Nếu không, gửi request của client lên server.
  - Nhận kết quả từ server và forward về cho client.
  - Khi có lỗi xảy ra thì đóng thread gây ra lỗi và xuất lỗi đó ra log.
  - Đây là "xương sống", là thành phần quan trọng nhất của chương trình proxy này.
- **Tham số:** socket interface của client cần xử lí và thông tin định danh của client đó (nhằm mục đích logging).
- **Trả về:** không.

```
def parse_header(request_content: bytes) → dict
```

- **Nhiệm vụ:** phân tích thông tin client gửi lên và trả về một dictionary là nội dung của các entity headers (nhằm mục đích logging và request forwarding).
- **Tham số:** thông tin client gửi lên.
- **Trả về:** một dictionary chứa `Method`, `URI`, các entity headers và nội dung của chúng.

```
def is_blocked(url: str) → bool
```

- **Nhiệm vụ:** kiểm tra một URL xem có bị block hay không (dữ liệu lấy từ file `blacklist.conf`).
- **Tham số:** URL cần kiểm tra.
- **Trả về:** một giá trị `boolean` cho biết URL có bị block hay không.

```
def get_target_info(header: dict) → dict
```

- **Nhiệm vụ:** phân tích các entity headers được phân tích bởi hàm `parse_header` ở trên nhằm lọc ra 2 thông tin quan trọng: `Host` và `Port` của server.
- **Tham số:** dictionary được trả về từ hàm `parse_header`.
- **Trả về:** 2 thông tin `Host` và `Port` của server đích.

```
def send_403_forbidden(client_socket: socket.socket) -> None
```

- **Nhiệm vụ:** gửi thông báo `HTTP/1.1 403 Forbidden` về client.
- **Tham số:** socket interface của client cần khước từ kết nối.
- **Trả về:** không.
- Vì trong file HTML gửi về có chèn hình ảnh, để tránh rắc rối khi phải xử lí thêm các request của client sau đó, chúng em sẽ chèn trực tiếp binary code của hình đó vào trong file HTML gửi về cho client.

```
def recvall(s: socket.socket) -> bytes
```

- **Nhiệm vụ:** nhận request từ client.
- **Tham số:** socket interface của client cần nhận request.
- **Trả về:** request của client dưới dạng raw bytes (không decode).

```
def log(s: str) -> None
```

- **Nhiệm vụ:** in ra thông tin log trên màn hình kèm theo ngày tháng năm và màu chữ (xanh nếu là thông báo bình thường, đỏ nếu lỗi, ...). Màu chữ chỉ xuất hiện nếu dùng các hệ điều hành tương tự Unix như Linux hay macOS. Trên Windows có thể sẽ xuất hiện các kí tự lạ nếu không có terminal emulator thích hợp.
- **Tham số:** chuỗi cần in.
- **Trả về:** không.

```
def req_log(headers: dict) -> None
```

- **Nhiệm vụ:** In ra một số thông tin quan trọng từ client gửi lên.
- **Tham số:** một dictionary chứa thông tin được phân tích và trả về bởi hàm `parse_header`.
- **Trả về:** không.

4. Chạy chương trình

- Chương trình được test trên nền tảng Linux. Một số tính năng như màu chữ có thể không khả dụng trên Windows nếu không có terminal emulator thích hợp.
- Do chỉ hỗ trợ HTTP, chúng em sẽ chuẩn bị sẵn 1 số trang web còn sử dụng giao thức này để test chương trình, cụ thể như sau:

STT	Tên	Thể loại	Link
1	Báo Bình Định	Tin tức	http://www.baobinhdinh.com.vn/
2	Báo Quảng Ngãi	Tin tức	http://baoquangngai.vn/
3	8086 assembler tutorial for beginners	Giáo dục	http://jbwyatt.com/253/emu/asm_tutorial_01.html
4	Cộng đồng Arduino Việt Nam	?	http://arduino.vn
5	Đại học Quy Nhơn	Giáo dục	http://www.qnu.edu.vn/

Cách chạy chương trình

Thầy/cô vui lòng cài đặt thư viện `termcolor` từ pip trước khi chạy chương trình này.

```
macOS/Linux
$ pip3 install termcolor
```

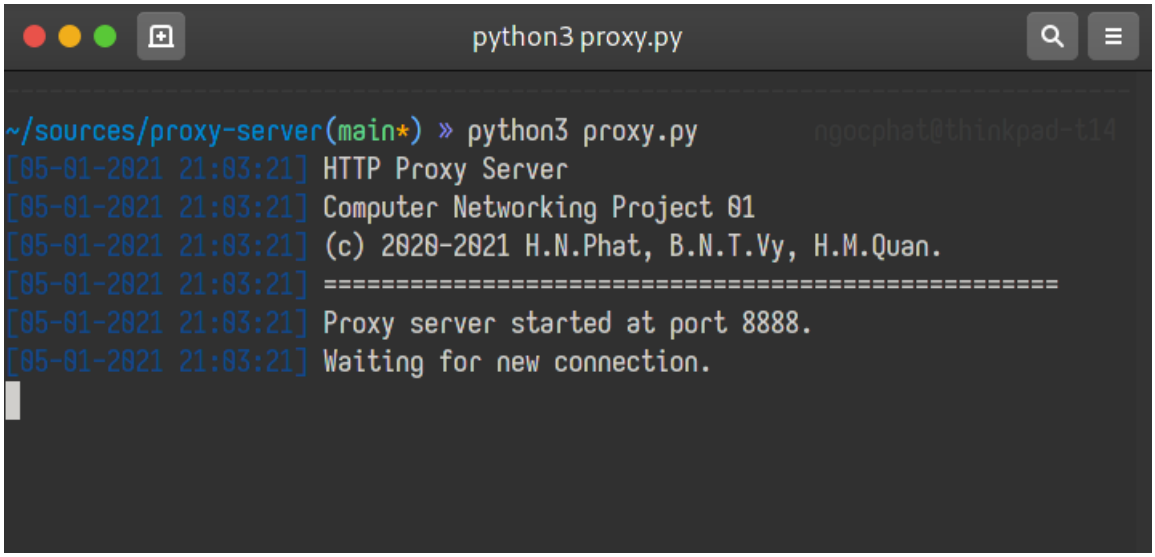
```
Windows
D:\....> pip install termcolor
```

Nếu là Linux/macOS xin thầy/cô gõ lệnh sau trong Terminal để chạy chương trình

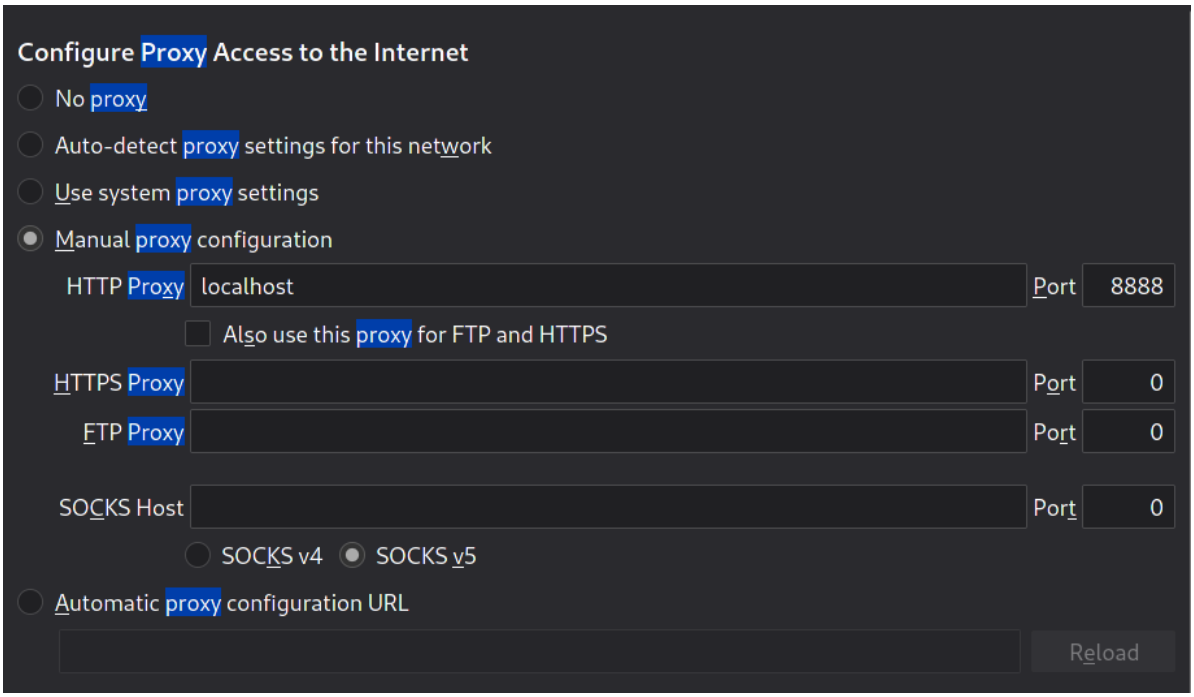
```
$ python3 proxy.py
```

Nếu thầy/cô chạy từ Windows, để có thể sử dụng đầy đủ nhất chương trình của chúng em (bao gồm tính năng hiển thị màu chữ), xin vui lòng sử dụng một terminal emulator thích hợp. Cách đơn giản nhất là sử dụng terminal được tích hợp bên trong Visual Studio Code.

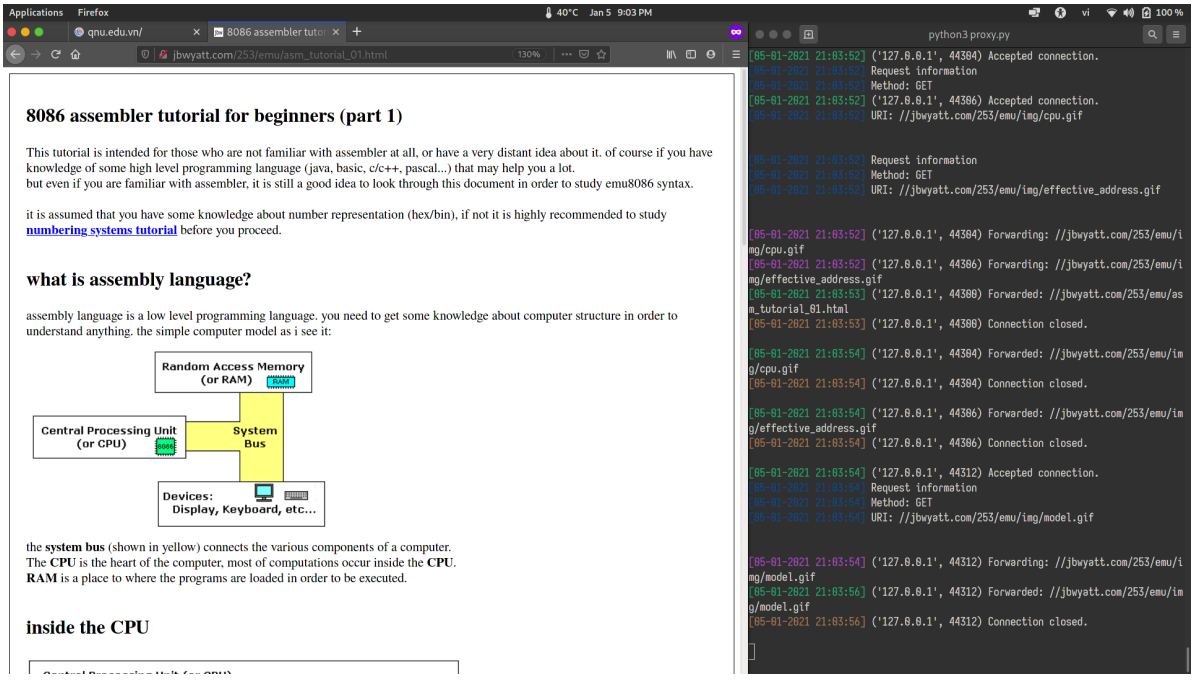
```
D:\....> python proxy.py
```



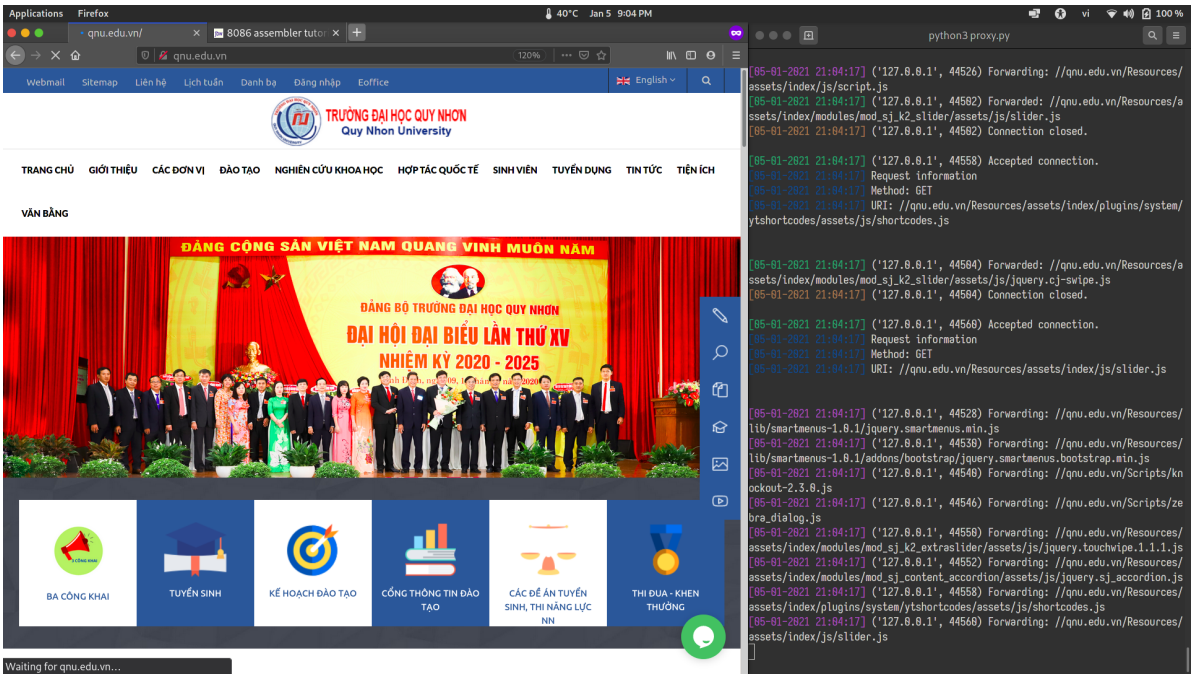
Khởi chạy từ shell trên Linux



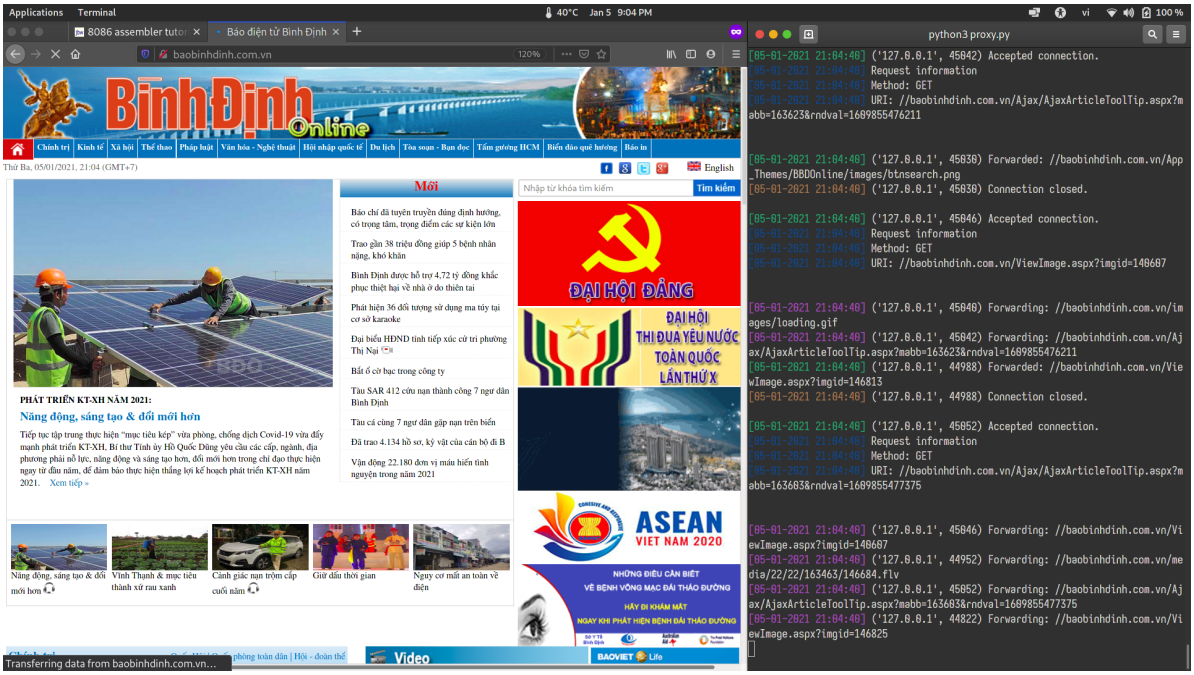
Thiết lập proxy server trên trình duyệt Firefox



Một trang web đơn giản, chỉ có plain text và một số hình ảnh nhẹ



Một trang web phức tạp, nhiều đối tượng Javascript.



Một trang web phức tạp, gồm nhiều đối tượng Javascript, ảnh động, video.

Trang HTML được trả về khi truy cập một trang web bị cấm

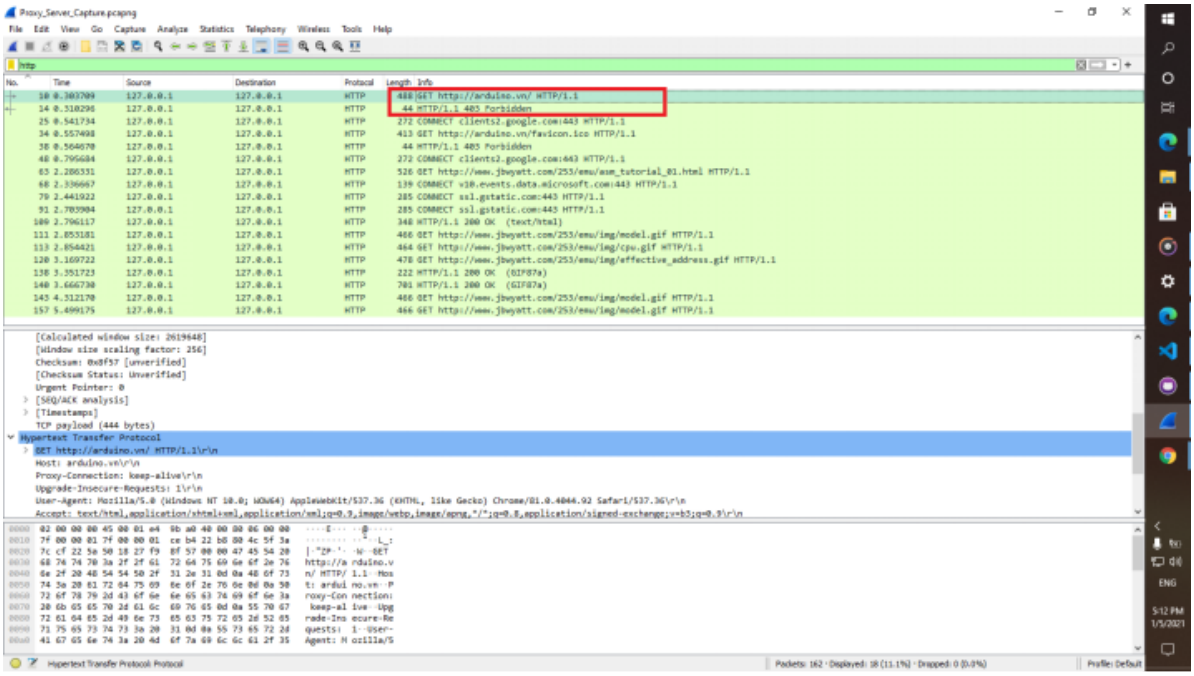
- Trong quá trình test không thể tránh khỏi những sai sót. Nếu trong quá trình chạy lỡ thầy/cô không chạy được ứng dụng proxy server của nhóm chúng em, thầy/cô có thể liên hệ nhóm chúng em qua email 19120615@student.hcmus.edu.vn (đại diện), hoặc có thể theo dõi video dưới đây do chúng em quay màn hình lại trên máy của mình (lúc đó proxy chạy hoàn toàn bình thường).

  https://drive.google.com/file/d/18cd5nNsHYdT-rSA0FgTy53sZGkvYqIta/view?usp=sharing

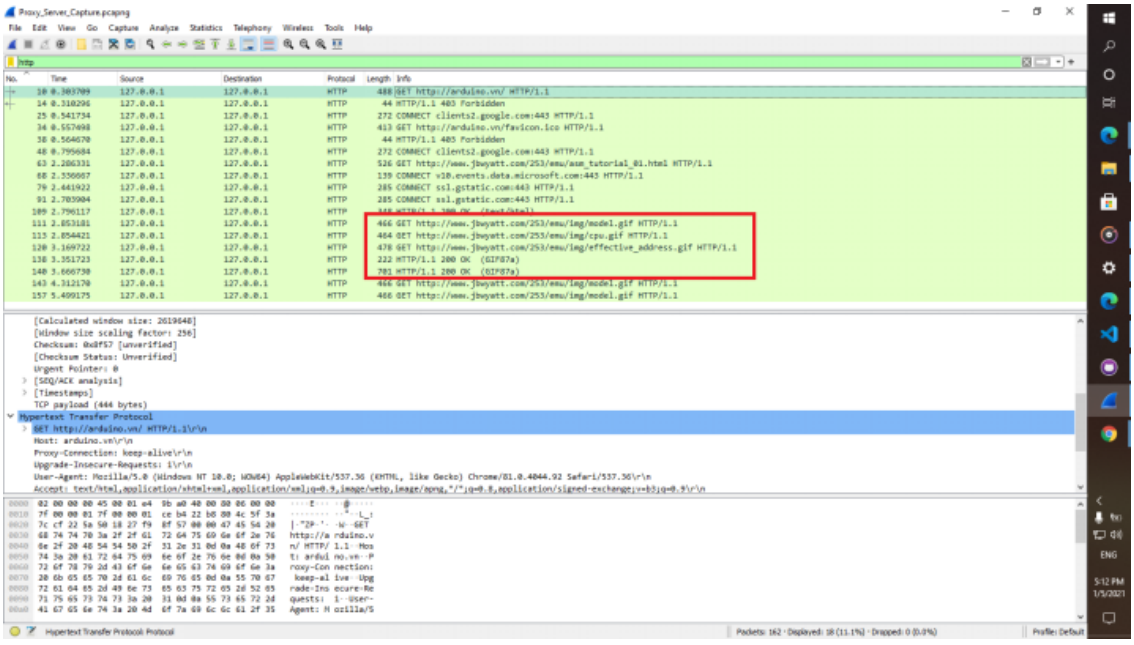
5. Những điều đã làm được/chưa làm được

- Hoạt động không ổn định lắm. Đôi khi web không load được phải refresh lại.
- Truy cập 2 trang cùng lúc sẽ làm mạng bị nghẽn trong 1 thời gian dù đã lập trình theo hướng multithreading.
- Tuy nhiên, nếu kết nối thành công, proxy hoạt động tốt với tất cả các trang mà chúng em đã liệt kê ở bên trên, kể cả các trang web khá nặng như web của Trường Đại học Quy Nhơn.
- Chưa thể hỗ trợ HTTPS (gần như bất khả thi khi lập trình với raw socket interface ở mức độ căn bản).
- Tốc độ còn khá chậm vì cần phải có một khoảng thời gian timeout là 1.5 giây khi nhận data từ server để gửi về client vì phải chắc chắn gói tin đã được truyền hết. Không thể nhận data từ server rồi gửi về client trong một lượt vì dữ liệu truyền bằng giao thức TCP là stream-based, không phải message-based. Do đó tụi em không nghĩ ra cách nào để biết được gói tin đã được truyền hết hay chưa.
  Nhược điểm của phương pháp này là nếu mạng bị nghẽn thì gói tin sẽ bị mất nếu hết 1.5s thời gian chờ. Tuy nhiên thông thường trường hợp này khó xảy ra vì proxy server này chỉ nhận 1 KB trong 1 chunk. Rất hiếm để một gói tin 1 KB không truyền đi hết trong 1.5s (tương đương tốc độ $\leq$ 0.67 KB/s).

6. Bắt gói tin từ Wireshark



- Với lệnh GET thứ nhất mà client gửi lên, proxy kiểm tra và thấy đường dẫn đó nằm trong danh sách đen của proxy, do đó proxy chặn luôn kết nối lên trang web đó và tặng cho client mã 403 Forbidden.



- Với lệnh GET thứ 2 mà client gửi lên, proxy kiểm tra và lần này đường dẫn đó không nằm trong danh sách đen của proxy, proxy lúc này sẽ gửi yêu cầu lên server. Sau đó server xử lý và trả về kết quả cho proxy, và proxy gửi kết quả xử lý của server về lại client (mã 200 OK).
- Tổng kết chung (cho mẫu proxy server theo yêu cầu đề bài):
    - Khi client muốn thiết lập kết nối đến server đích (gọi tắt: server), nó phải đi qua proxy server (gọi tắt: proxy) trước.
    - Đầu tiên, proxy sẽ được chạy trên localhost, lắng nghe ở port 8888.
    - Một client nào đó thiết lập kết nối đến proxy.
    - Client gửi request lên, proxy mở một thread mới để handle request đó. Trong lúc đó nó sẽ lắng nghe chờ các kết nối khác. Nếu nhận một request khác được nó cũng tiến hành các bước dưới đây một cách đồng thời (song song).
    - Sau khi nhận được hoàn chỉnh request, proxy sẽ tiến hành parse và check thông tin đầu tiên là `Method` xem có phải là `GET` hoặc `POST` hay không. Nếu không thì gửi về `200 OK`, đồng thời ngắt kết nối với client.

- Proxy tiếp theo tiến hành kiểm tra URL/hostname xem thử có nằm trong danh sách chặn hay không.
- Nếu có, proxy gửi về mã lỗi `403`, đồng thời chấm dứt làm ăn với client.
- Nếu không, proxy tiếp tục gửi request đó lên server.
- Proxy chờ server trả lời và chủ động forward nguyên vẹn thông tin đó xuống client (nếu không có lỗi xảy ra giữa chừng).
- Sau khi client nhận được thông tin mà mình cần, 2 bên tiến hành đóng kết nối.

7. Những tình huống cần proxy server trong thực tế

- Khi một tổ chức muốn kiểm soát kết nối từ bên trong ra bên ngoài. Ví dụ như:
  - Nhà mạng muốn kiểm soát các trang web mà người dùng của mình truy cập.
  - Công ty muốn kiểm soát nội dung mà nhân viên của mình truy cập.
  - Làm portal login page ở các sân bay, ký túc xá, trường học (ví dụ như Meganet ở trường ĐH KHTN, INET ở KTX ĐHQG, ...).
- Khi một tổ chức muốn kiểm soát kết nối từ bên ngoài vào bên trong. Ví dụ như:
  - Một công ty muốn thuê "vệ sĩ" điều tiết lưu lượng mạng để chống DDoS, điển hình là cloudfare.
- Và nhiều tình huống khác.

Tài liệu tham khảo

1. https://docs.python.org/3/library/socket.html
2. https://www.geeksforgeeks.org/proxy-server/
3. https://pypi.org/project/termcolor/
4. https://stackoverflow.com/questions/2429934/is-it-possible-to-put-binary-image-data-into-html-markup-and-then-get-the-image
5. Một số tài liệu khác từ Internet.

--- HẾT ---