HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN

BÁO CÁO BÀI THỰC HÀNH
HỌC PHẦN: THỰC TẬP CƠ SỞ
MÃ HỌC PHẦN: INT13147

BÀI THỰC HÀNH 4.1
LẬP TRÌNH CLIENT/SERVER
ĐỂ TRAO ĐỔI THÔNG TIN AN TOÀN

Sinh viên thực hiện:

B22DCAT018 – Nguyễn Hoàng Anh

Giảng viên hướng dẫn: ThS. Ninh Thị Thu Trang

HỌC KỲ 2 NĂM HỌC 2024-2025

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

CHƯƠNG 1. TÌM HIỂU LÝ THUYẾT

1.1 Mục đích

Hiểu về cơ chế client/server và có thể tự lập trình client/server dựa trên socket, sau đó thực hiện ca đặt giao thức đơn giản để trao đổi thông tin an toàn.

1.2 Tìm hiểu lý thuyết

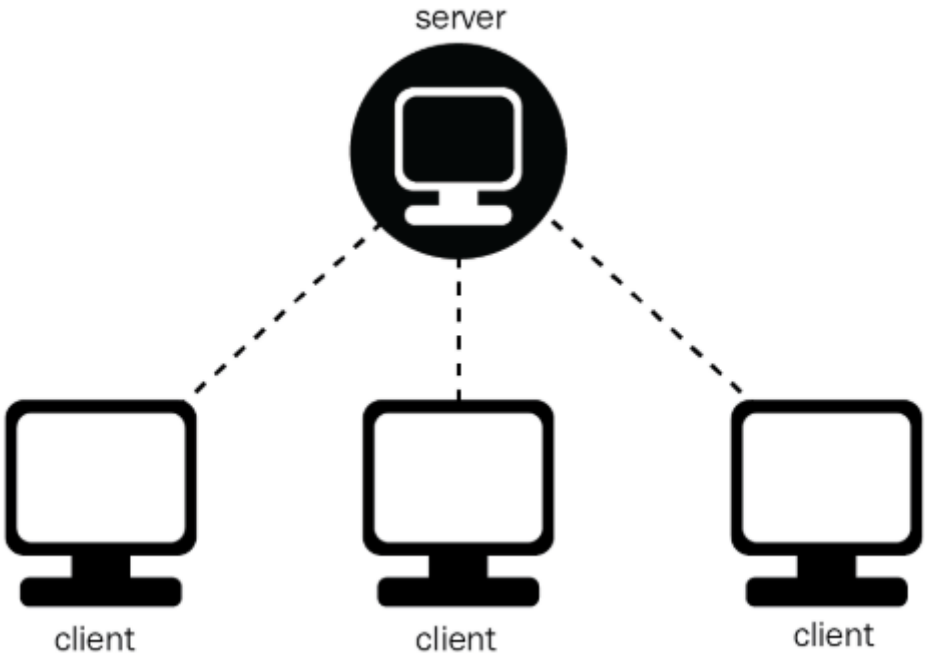
1.2.1 Cơ chế Client/Server

1.2.1.1 Giới thiệu về cơ chế Client/Server

Cơ chế Client/Server bản chất là 2 máy tính (client và server) giao tiếp và truyền tải dữ liệu cho nhau:

- Máy tính đóng vai trò là máy khách (Client): Với vai trò là máy khách, chúng sẽ không cung cấp tài nguyên đến các máy tính khác mà chỉ sử dụng tài nguyên được cung cấp từ máy chủ. Một client trong mô hình này có thể là một server cho mô hình khác, tùy thuộc vào nhu cầu sử dụng của người dùng.
- Máy tính đóng vai trò là máy chủ (Server): Là máy tính có khả năng cung cấp tài nguyên và các dịch vụ đến các máy khách khác trong hệ thống mạng. Server đóng vai trò hỗ trợ cho các hoạt động trên máy khách client diễn ra hiệu quả hơn.

Mô hình Client Server là mô hình mạng máy tính trong đó các máy tính con được đóng vai trò như một máy khách, chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ. Để máy chủ xử lý yêu cầu và trả kết quả về cho máy khách đó.

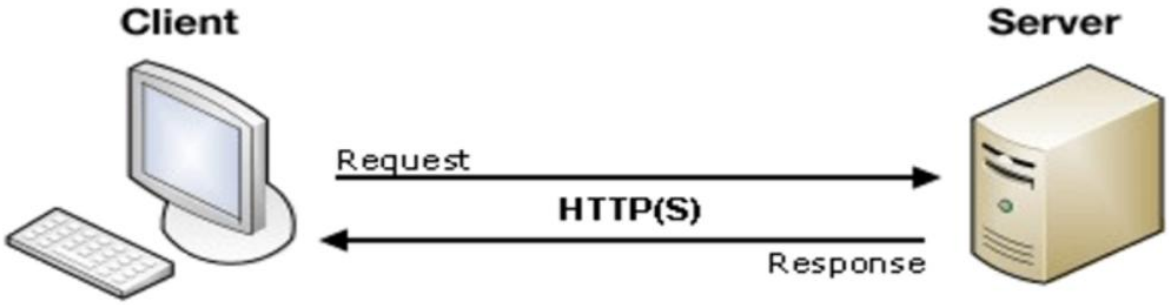


Hình 1 - mô hình Client/Server

1.2.1.2 Nguyên tắc hoạt động của mô hình Client/Server:

Trong mô hình Client Server, server chấp nhận tất cả các yêu cầu hợp lệ từ mọi nơi khác nhau trên Internet, sau đó trả kết quả về máy tính đã gửi yêu cầu đó

Máy tính được coi là máy khách khi chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ và đợi câu trả lời được gửi về.



Hình 2 - nguyên tắc hoạt động của mô hình client/server

Để máy khách và máy chủ có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn nhất định, và chuẩn đó được gọi là giao thức. Một số giao thức được sử dụng phổ biến hiện nay như: HTTPS, TCP/IP, FTP,...

Nếu máy khách muốn lấy được thông tin từ máy chủ, chúng phải tuân theo một giao thức mà máy chủ đó đưa ra. Nếu yêu cầu đó được chấp nhận thì máy chủ sẽ thu thập

thông tin và trả về kết quả cho máy khách yêu cầu. Bởi vì Server - máy chủ luôn luôn trong trạng thái sẵn sàng để nhận request từ client nên chỉ cần client gửi yêu cầu tín hiệu và chấp nhận yêu cầu đó thì server sẽ trả kết quả về phía client trong thời gian ngắn nhất.

1.2.1.3 Ưu điểm và nhược điểm của cơ chế Client/Server

Ưu điểm:

- Giúp chúng ta có thể làm việc trên bất kì một máy tính nào có hỗ trợ giao thức truyền thông. Giao thức chuẩn này cũng giúp các nhà sản xuất tích hợp lên nhiều sản phẩm khác nhau mà không gặp phải khó khăn gì.
- Có thể có nhiều server cùng làm một dịch vụ, chúng có thể nằm trên nhiều máy tính hoặc một máy tính.
- Chỉ mang đặc điểm của phần mềm mà không hề liên quan đến phần cứng, ngoài yêu cầu duy nhất là server phải có cấu hình cao hơn các client.
- Hỗ trợ người dùng nhiều dịch vụ đa dạng và sự tiện dụng bởi khả năng truy cập từ xa.
- Cung cấp một nền tảng lý tưởng, cho phép cung cấp tích hợp các kỹ thuật hiện đại như mô hình thiết kế hướng đối tượng, hệ chuyên gia, hệ thông tin địa lý (GIS).

Nhược điểm:

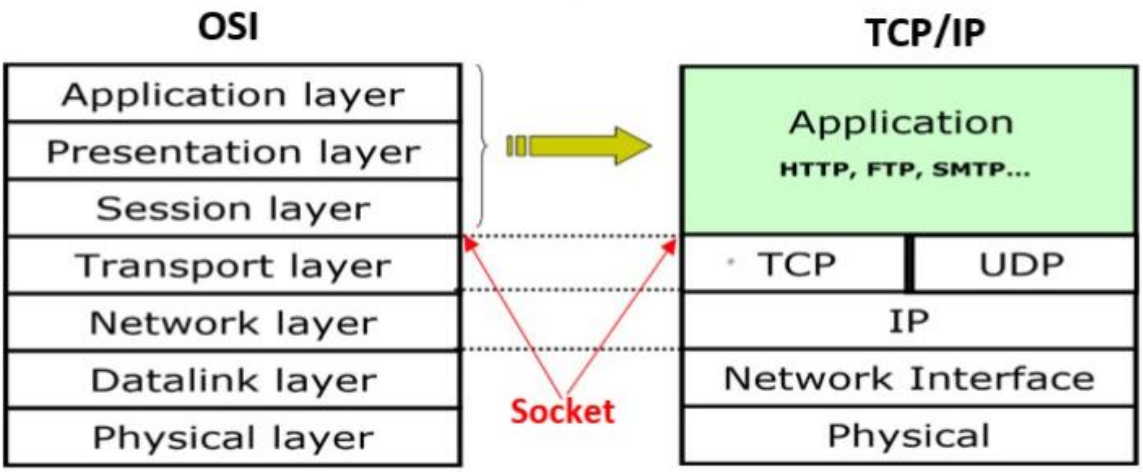
Vấn đề bảo mật dữ liệu thông tin đôi khi còn chưa được an toàn lắm. Vì do phải trao đổi dữ liệu giữa 2 máy tính khác nhau ở 2 khu vực địa lý cách xa nhau. Và đây cũng nhược điểm duy nhất của mô hình này.

Tuy nhiên vấn đề này thì có một số giao thức đã hỗ trợ bảo mật dữ liệu khi truyền tải. Giao thức được sử dụng phổ biến như HTTPS.

1.2.2 Lập trình socket với TCP

1.2.2.1 Tổng quan về socket

Trong hệ thống mạng máy tính tồn tại những mô hình tham chiếu có kiến trúc phần tầng (OSI, TCP/IP…) nhằm hỗ trợ chức năng trao đôi thông tin giữa các ứng dụng ở nhiều máy tính khác nhau.



Hình 3 - mô hình OSI và TCP/IP

Dữ liệu bên gửi sẽ được đóng gói (Encapsulation) từ tầng trên đến tầng cuối là tầng vật lí (Physical Layer), sau đó nhờ tầng vật lí này chuyển dữ liệu đến tầng vật lí máy bên nhận, bên nhận tiến hành giải mã (decapsulation) gói dữ kiện từ tầng dưới lên tầng trên cùng, là tầng ứng dụng (application layer).

Ở đây, Socket chính là cửa giao tiếp giữa tầng ứng dụng và tầng giao vận (Transport layer). Nói cách khác, Socket là giao diện do ứng dụng tạo ra trên máy trạm, quản lí bởi hệ điều hành qua đó các ứng dụng có thể gửi/nhận thông điệp đến/từ các ứng dụng khác. Ở đó, Socket sẽ được ràng buộc với một mã số cổng (Port Number) để giúp tầng giao vận định danh được ứng dụng nhận/gửi thông điệp.

Có thể thấy ở hình ảnh trên, tầng giao vận có 2 phương thức là TCP (Ttransmission Control Protocol) và UDP (User Datagram Protocol), như vậy socket cơ bản là có 2 loại: Stream Socket sử dụng TCP truyền dòng bytes và Datagram Socket dử dụng UDP truyền gói tin. Với ngôn ngữ lập trình Java, chúng ta được cung cấp 3 loại khác nhau của sockets:

- Stream Socket (TCP) : Tạo luồng dữ liệu hai chiều, đáng tin cậy, có trình tự và không trùng lặp, dữ liệu chỉ được gửi/nhận khi có đã có liên kết. Dùng với Socket Class của java.
- Datagram Socket (UDP): Có thể nhận dữ liệu không theo tình tự, trùng lặp. Dùng với DatagramSocket Class.
- Multicast Socket : cho phép dữ liệu được gửi đến nhiều bên nhận một lúc. Dùng với DatagramSocket Class.

1.2.2.2 Lập trình TCP socket

Đúng như tính chất của TCP chúng ta cần có liên kết 2 chiều trước khi server và client có thể trao đổi thông điệp với nhau.

Ban đầu, phía server tạo Socket được ràng buộc với một cổng (port number) để chờ nhận yêu cầu từ phía client.

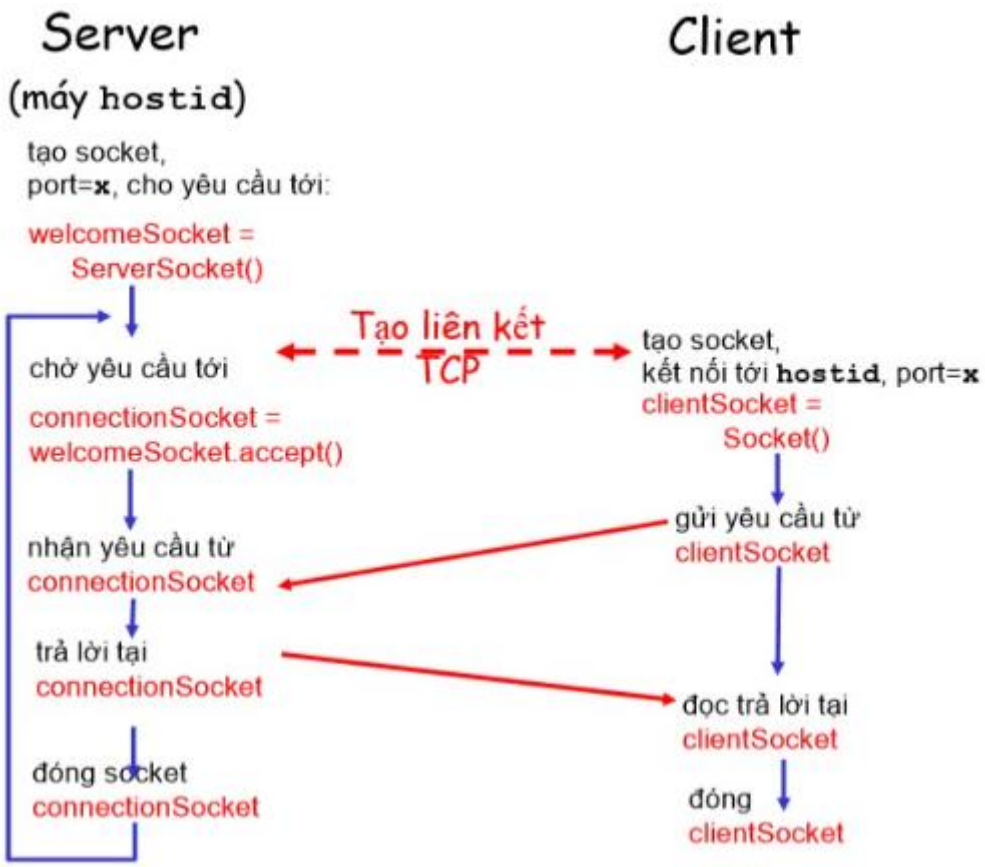
Tiếp đến phía client yêu cầu server bằng cách tạo một Socket TCP trên máy kèm với địa chỉ IP và port number của tiến tình tương ứng trên máy server. Khi client tạo Socket, client TCP tạo liên kết với server TCP và chờ chấp nhận kết nối từ server.

TCP cung cấp dịch vụ truyền dòng tin cậy và có thứ tự giữa client và server, giữa máy chủ và máy nhận chỉ có 1 địa chỉ IP duy nhất. Thêm vào đó, mỗi thông điệp truyền đi đều có xác nhận trả về.

Sau đây là một ví dụ ứng dụng đơn giản về lập trình TCP Socket.

Miêu tả ứng dụng:

- Client đọc dòng văn bản nhập từ bàn phím người dùng , gửi tới server qua Socket
- Server đọc các dòng văn bản gửi từ Socket
- Server sẽ chuyển lại dòng văn bản kèm theo “Server accepted” tới phía client qua Socket
- Client đọc dòng văn bản từ socket và in ra dòng văn bản nhận được từ server



Hình 4 - ví dụ cơ bản

Chúng ta có thể thấy rằng mỗi phía server và client đều có 2 luồng dữ liệu, một luồng ra Socket để gửi thông điệp và một luồng vào từ Socket để nhận thông điệp, như vậy với mỗi bên mình có hai biến input và output (inFromServer, outToServer và inFromClient, outToClient).

CHƯƠNG 2. NỘI DUNG THỰC HÀNH

2.1 Chuẩn bị môi trường

Môi trường Python để chạy ứng dụng client/server

Phần mềm Wireshark

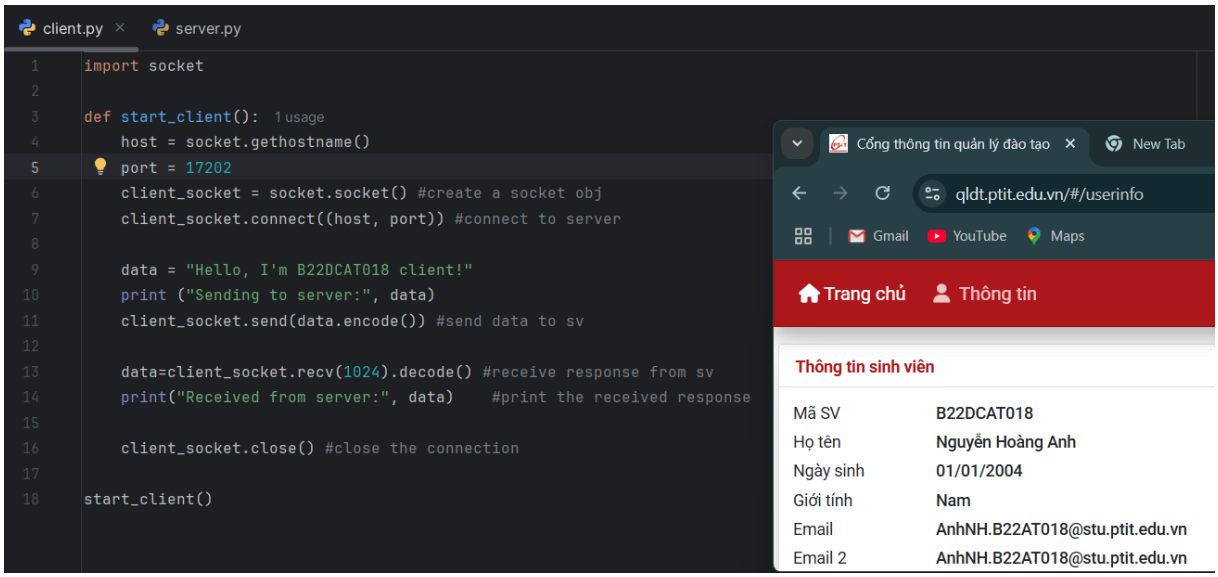
2.2 Các bước thực hiện

2.2.1 Lập trình client và server với TCP socket

Lập trình client

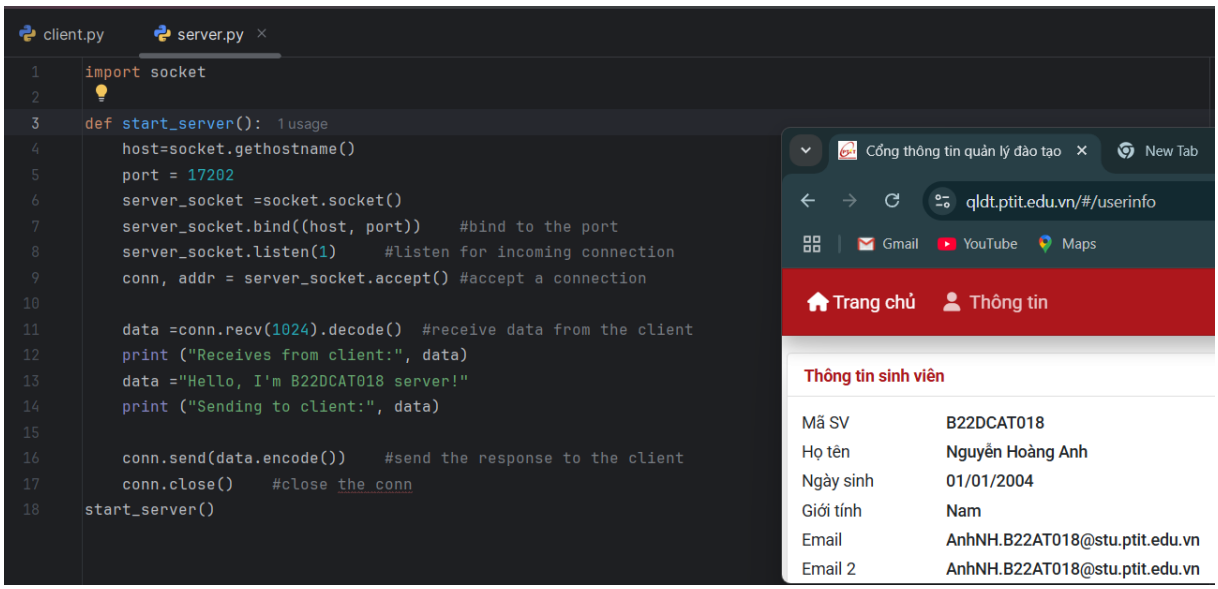
Host = socket.gethostname() → lấy địa chỉ ip máy tính đang chạy chương trình

Port = 17202 → cổng được sử dụng để kết nối (chọn port sao cho không trùng với port đang chạy dịch vụ trên máy)



Hình 5 - lập trình client

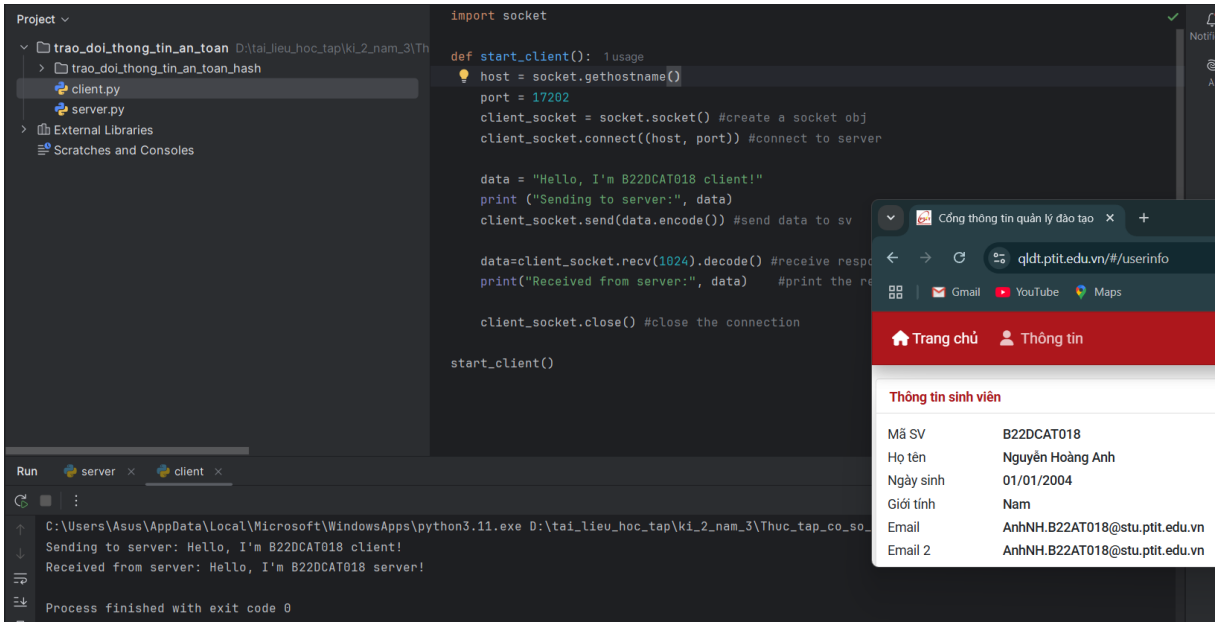
Lập trình server



Hình 6 - lập trình server

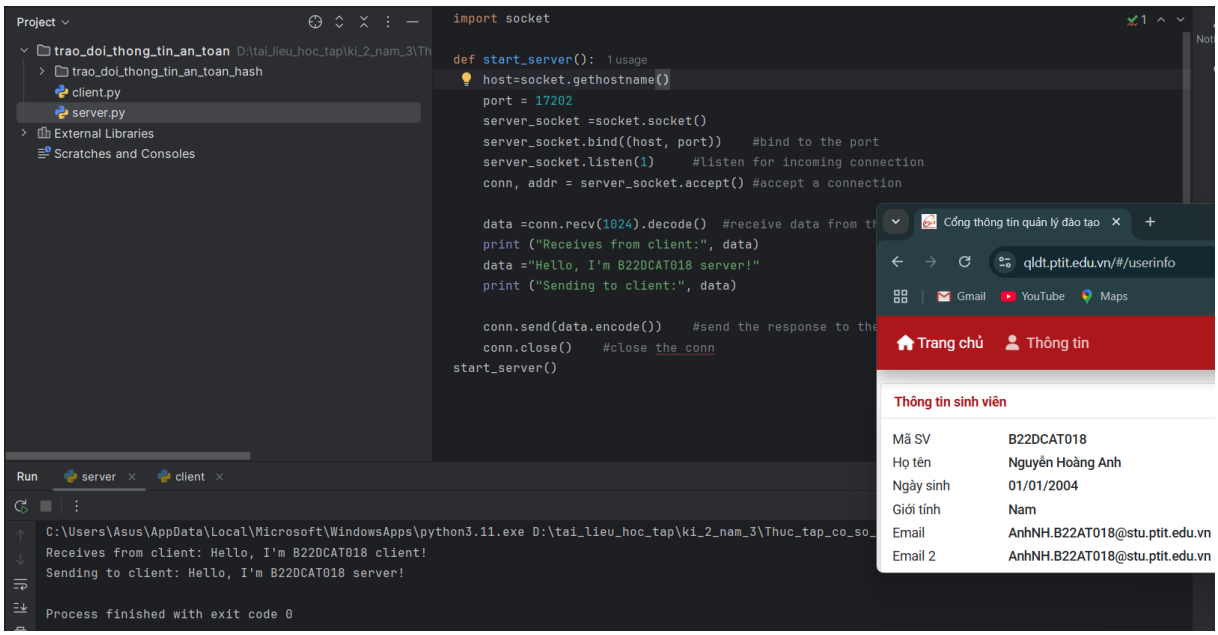
Chạy server sau đó chạy client

Client gửi thông điệp cá nhân hóa cho server: “Hello, I am <mã sinh viên> client.”



Hình 7 - client gửi thông điệp cho server và nhận lại thông điệp từ server

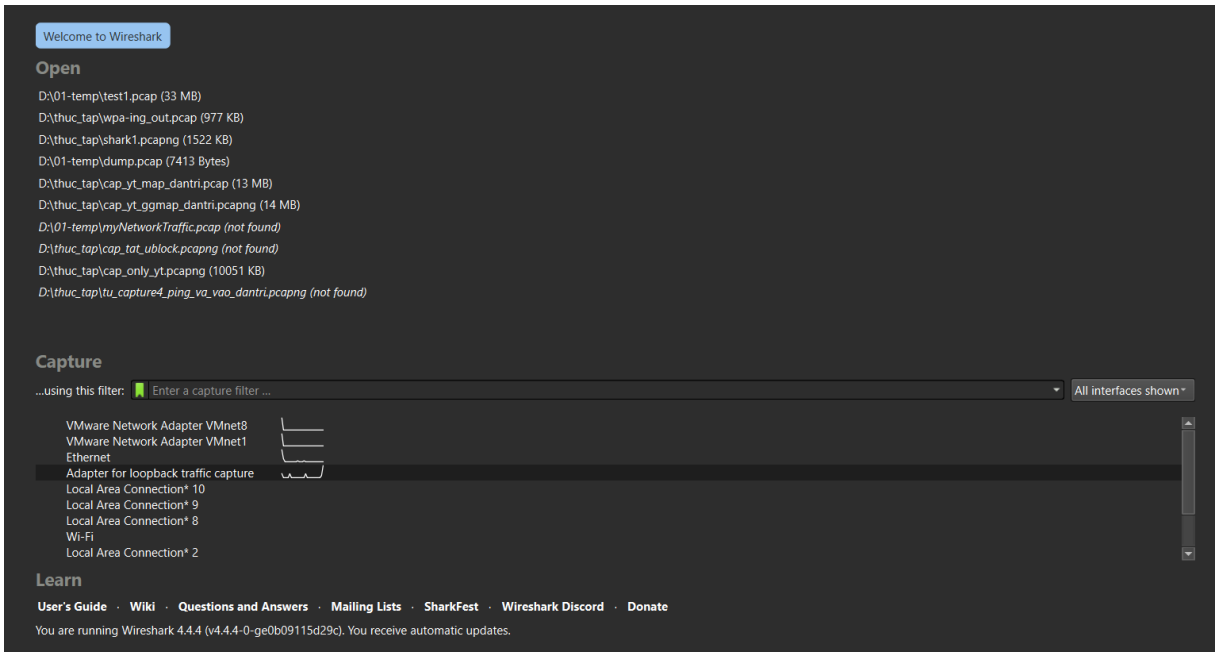
Server nhận được hiển thị thông điệp nhận được và gửi lại client thông điệp: server gửi lại “Hello, I am <mã sinh viên> server”



Hình 8 - server nhận thông điệp từ client và gửi lại cho client

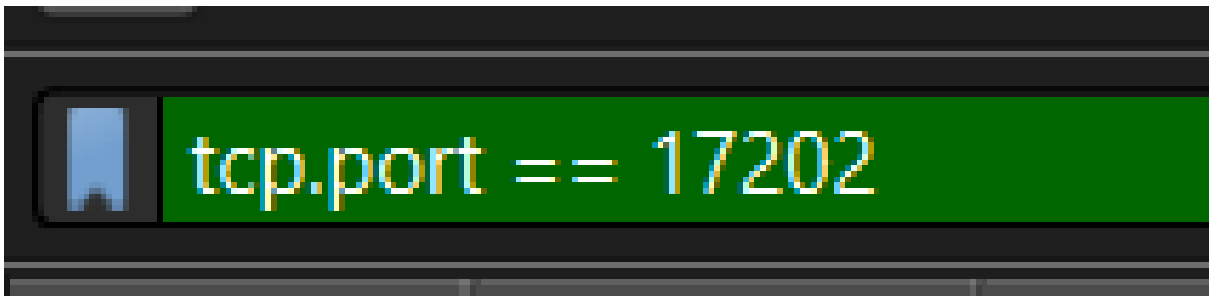
Sử dụng Wireshark để bắt các thông tin đã gửi từ client đến server và ngược lại

Chọn interface: Adapter for loopback traffic capture



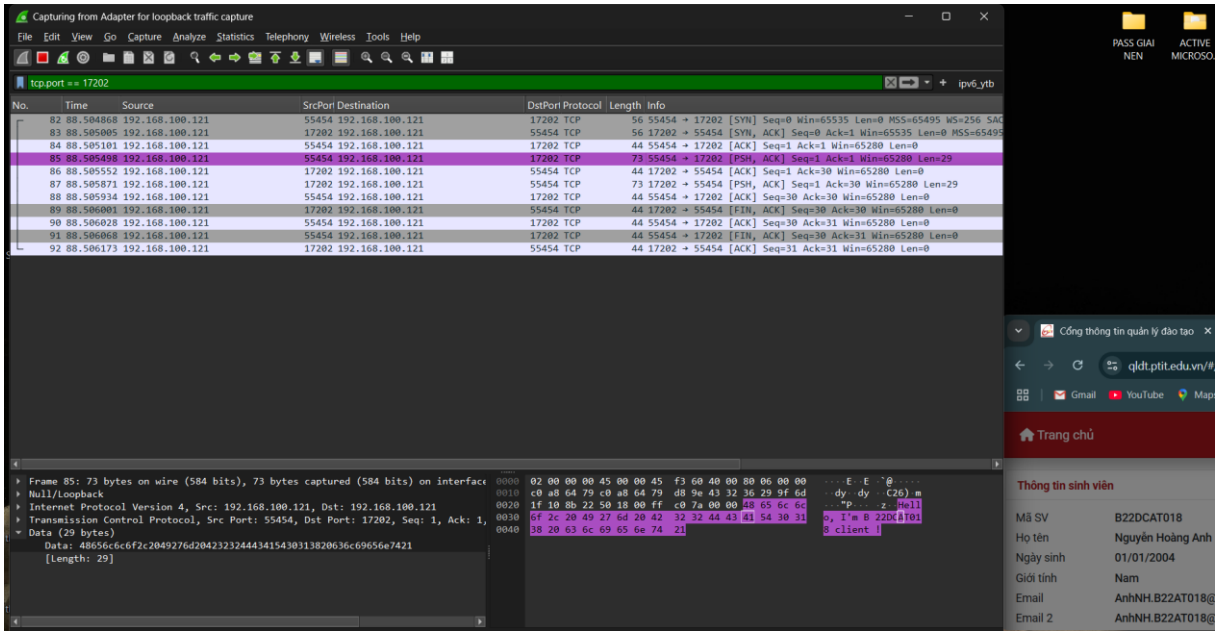
Hình 9 - mở wireshark và chọn interface

Lọc theo filter: tcp.port = 17202



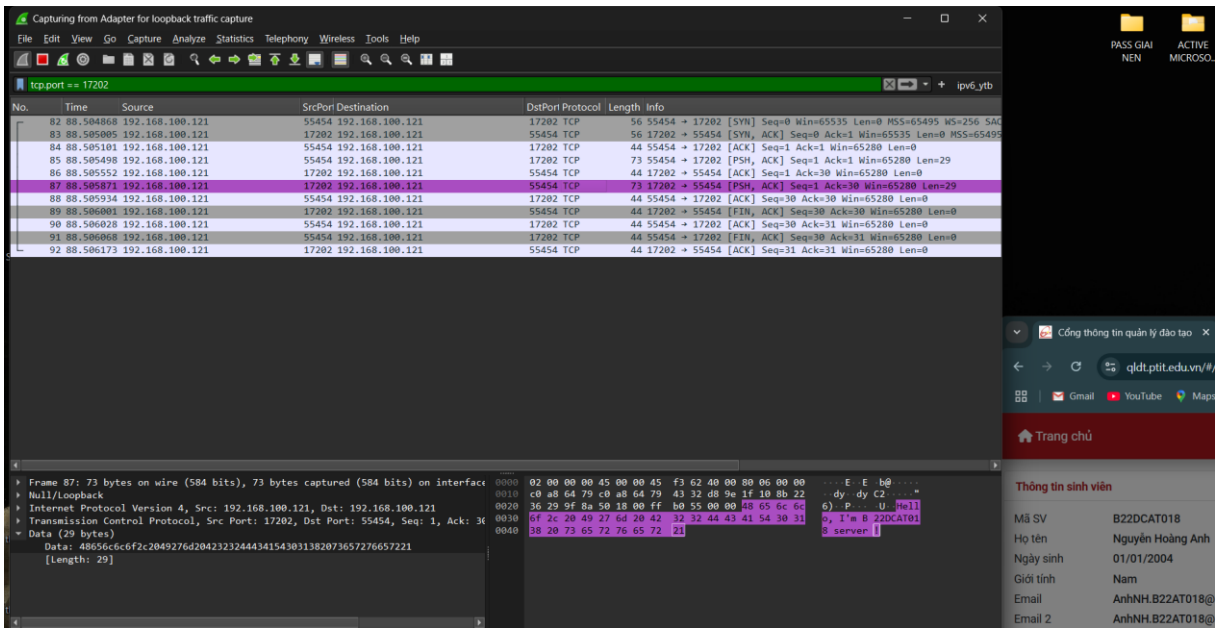
Hình 10 - lọc theo port

Bắt được gói tin chứa thông điệp: “Hello, I’m B22DCAT018 client!” của client (port 55454) gửi đến server (port 17202)



Hình 11 - bắt được gói tin chứa thông điệp của client

Bắt được gói tin chứa thông điệp “Hello, I’m B22DCAT018 server!” của server (port 17202) gửi đến client (port 55454)



Hình 12 - bắt được gói tin chứa thông điệp của server

2.2.2 Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi

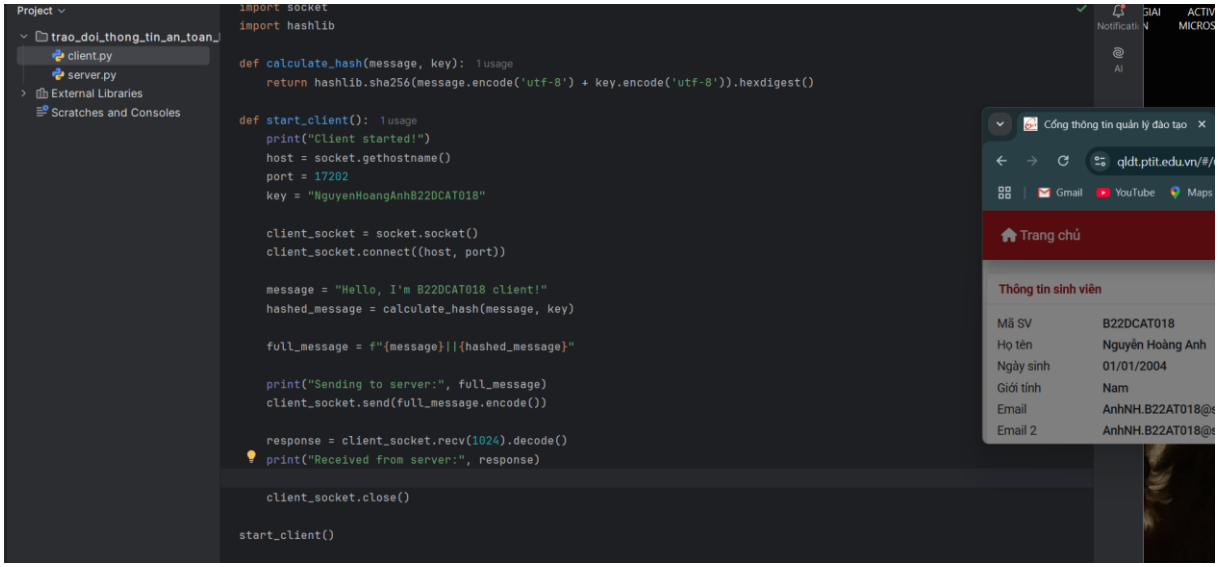
Từ client và server, sửa đổi để sao cho: khi gửi thông điệp sẽ gửi kèm theo giá trị băm của (thông điệp + key) để phía bên kia kiểm tra xác minh tính toàn vẹn. Hai bên có thể thống nhất một giá trị key trước đó.

Lập trình server với key

Hình 13 - lập trình server với key (1)

Hình 14 - lập trình server với key (2)

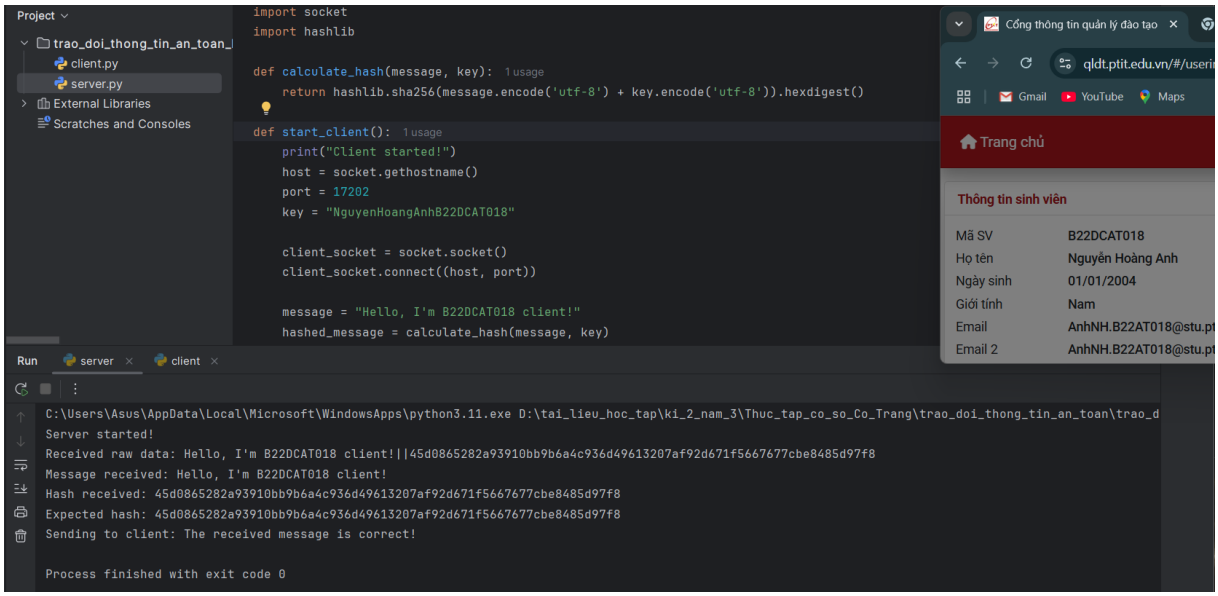
Lập trình client với key



Hình 15 - lập trình client với key

Khởi chạy server và client

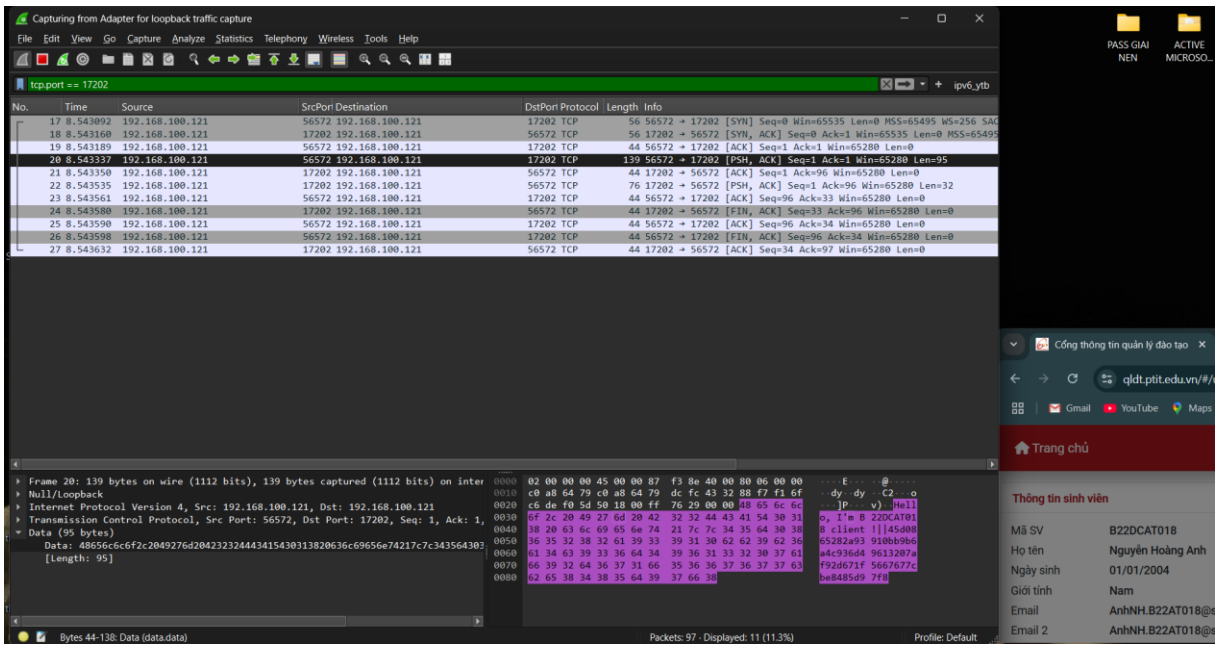
Hình 16 - client gửi thông điệp cùng mã hash cho server



Hình 17 - server so sánh mã hash và gửi lại thông điệp cho client

➔ Cả client và server đều cùng nhận được 1 mã hash do có 2 key giống nhau

Sử dụng wireshark để bắt gói tin

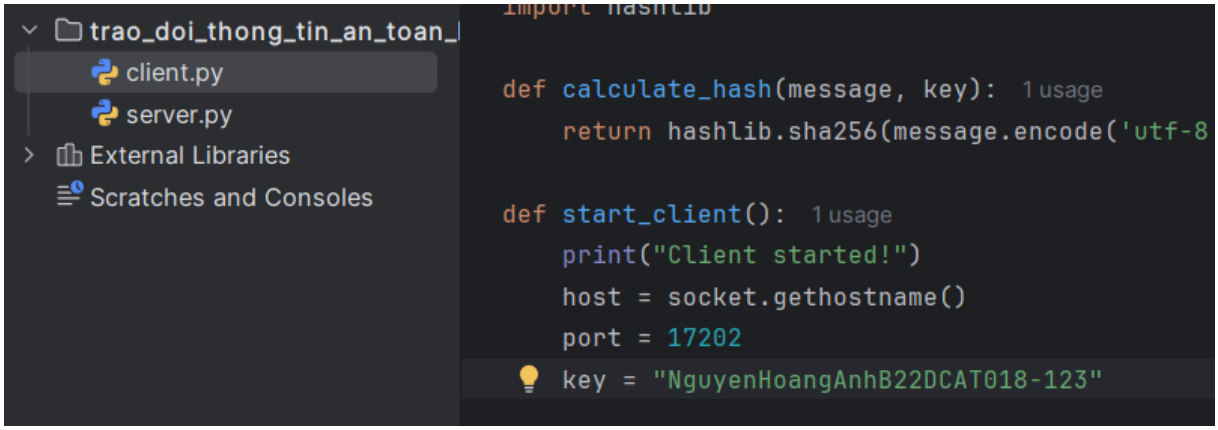


Hình 18 - dùng wireshark bắt được gói tin chứa thông điệp

➔ Wireshark bắt được gói tin có thông điệp và mã hash được gửi

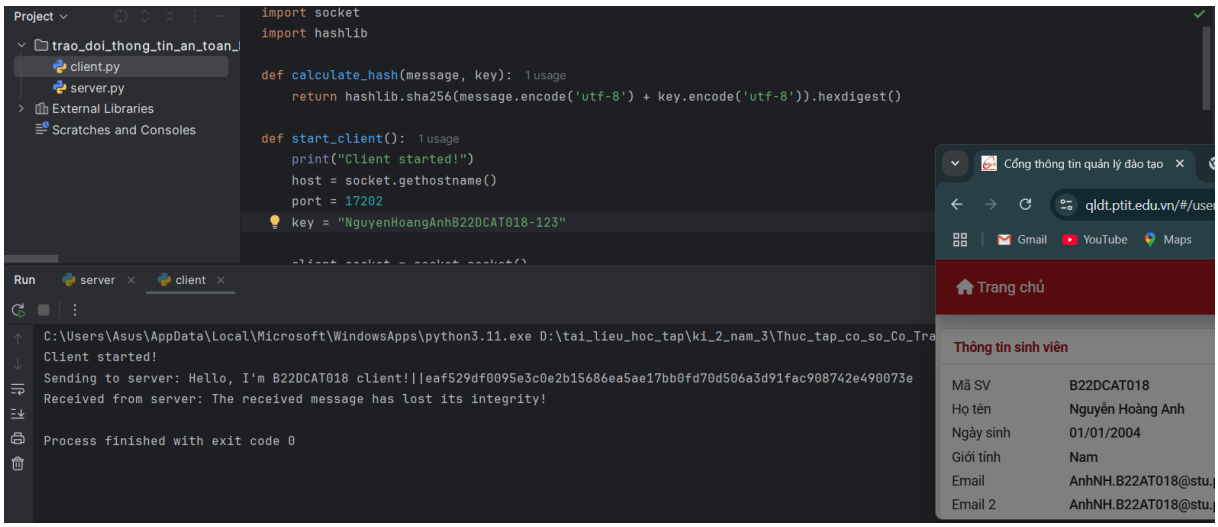
Thay đổi giá trị key tại client và thực hiện gửi lại, nếu không đáp ứng tính toàn vẹn cần thông báo: “The received message has lost its integrity.”

Thay đổi giá trị key tại client: từ “NguyenHoangAnhB22DCAT018” thành “NguyenHoangAnhB22DCAT018-123”



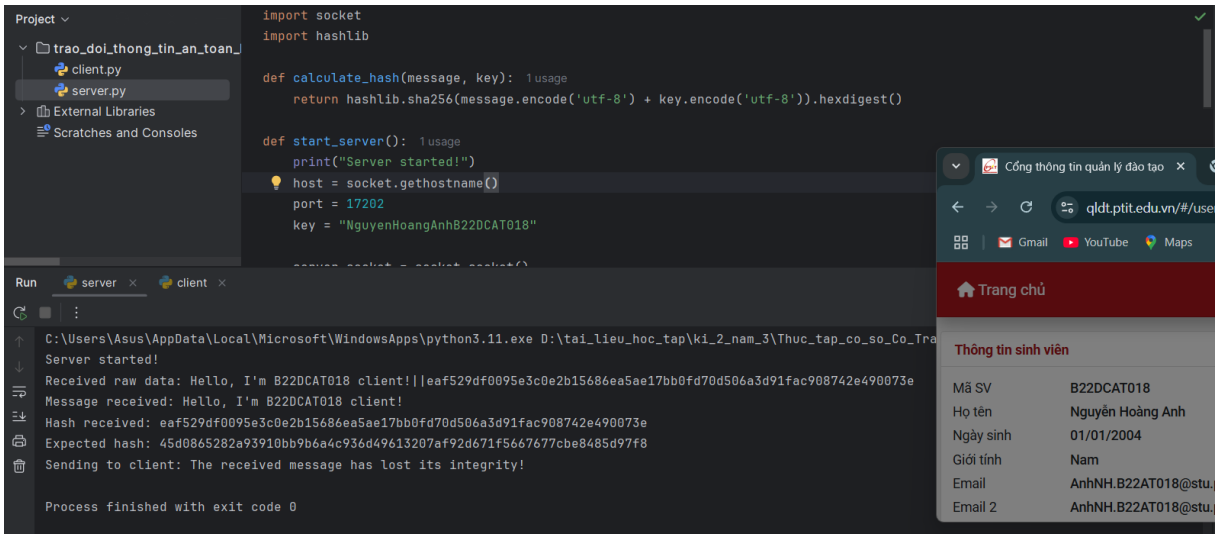
Hình 19 - thay đổi giá trị key của client

Client truyền thông điệp đi



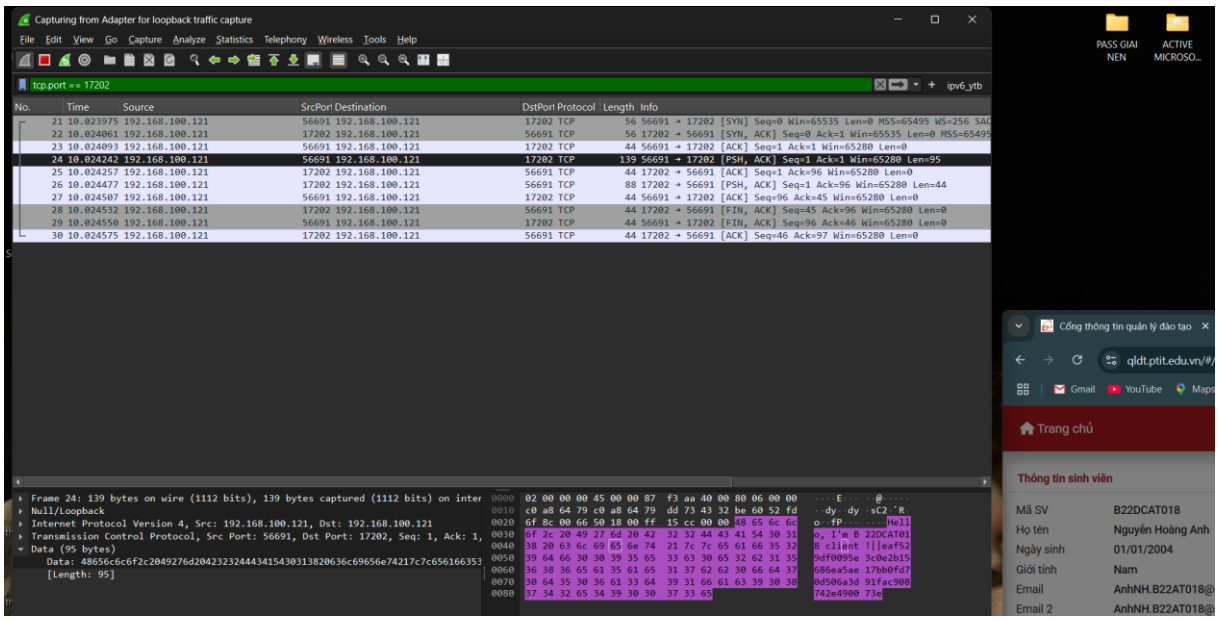
Hình 20 - client truyền thông điệp đến server

Server nhận được thông điệp, thấy hash đã thay đổi nên gửi đến client “The received message has lost its integrity”

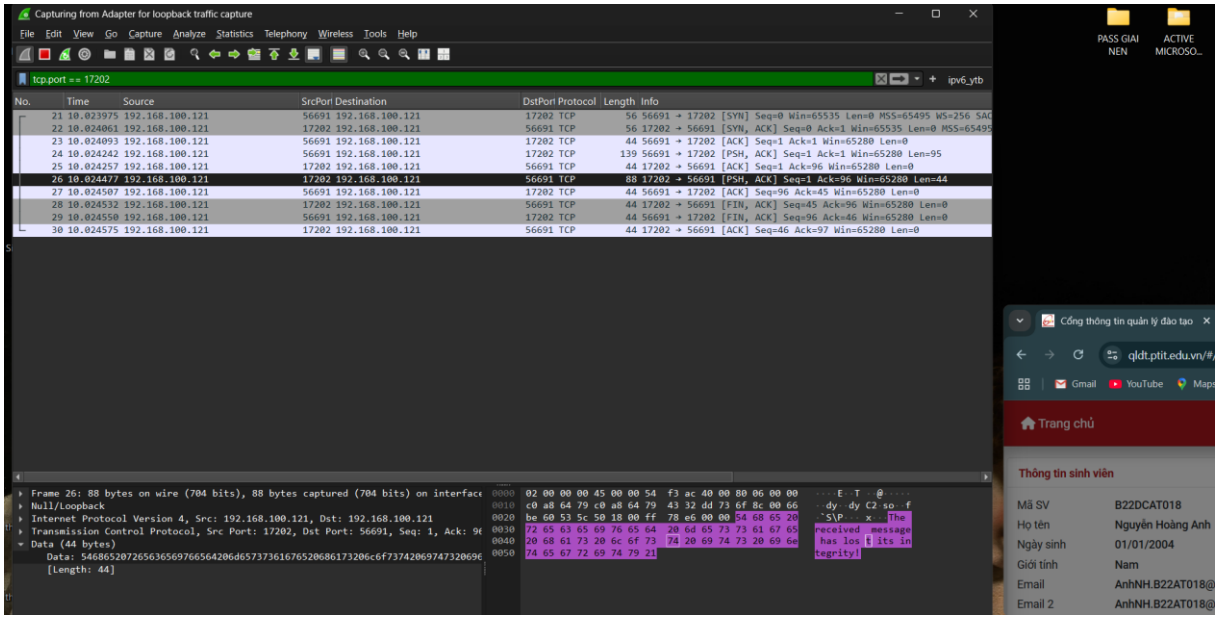


Hình 21 - server gửi lại kết quả mất tính toàn vẹn

Bắt được các bản tin trao đổi giữa client và server trong Wireshark



Hình 22 - dùng wireshark bắt gói tin chứa thông điệp của client cùng mã hash



Hình 23 - gói tin chứa kết quả mà server gửi lại cho client

TỔNG KẾT

Qua bài thực hành này, em đã nắm được các kiến thức và kỹ năng cơ bản về:

- Cơ chế lập trình client/server dựa trên socket tcp trong python.
- Thiết lập kết nối tcp giữa client và server: server lắng nghe (bind, listen), client chủ động kết nối (connect).
- Trao đổi thông tin đơn giản giữa client và server, bao gồm việc gửi và nhận các thông điệp.
- Đảm bảo tính toàn vẹn dữ liệu khi trao đổi thông tin bằng cách:
    - Tính toán giá trị hash (sha-256) của thông điệp + key bí mật tại bên gửi.
    - Kiểm tra giá trị hash tại bên nhận để phát hiện nếu thông điệp bị thay đổi.
- Bắt gói tin bằng wireshark để quan sát luồng dữ liệu trao đổi thực tế qua mạng.

Thông qua bài thực hành, em hiểu rõ hơn về ý nghĩa của bảo vệ tính toàn vẹn dữ liệu, một yếu tố cơ bản nhưng rất quan trọng trong việc xây dựng các hệ thống truyền thông an toàn.

TÀI LIỆU THAM KHẢO

1. Đinh Trường Duy, Phạm Hoàng Duy, Bài giảng Hệ điều hành Windows và Linux/Unix, Học viện Công Nghệ Bưu Chính Viễn Thông, 2022.
2. Tom Carpenter, Microsoft Windows Server Operating System Essentials, Sybex, 2011.
3. Chapter 2: Application Layer V8.1 (9/2020) tại địa chỉ http://gaia.cs.umass.edu/kurose_ross/ppt.php