

**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:

B22DCAT063 Lê Tiến Dương

Giảng viên hướng dẫn: PGS. TS. Hoàng Xuân Dậu

HỌC KỲ 2 NĂM HỌC 2024-2025

MỤC LỤC

MỤC LỤC	2
DANH MỤC CÁC HÌNH VẼ.....	3
DANH MỤC CÁC TỪ VIẾT TẮT.....	4
CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH	5
1.1 Mục đích.....	5
1.2 Tìm hiểu lý thuyết	5
1.2.1 Socket là gì?	5
1.2.2 Tại sao người dùng lại cần dùng đến socket?.....	5
1.2.3 Điều kiện để socket hoạt động?.....	6
1.2.4 Phân loại socket.....	7
1.2.5 Lập trình socket với TCP/IP	7
CHƯƠNG 2. NỘI DUNG THỰC HÀNH	9
2.1 Chuẩn bị môi trường	9
2.2 Các bước thực hiện.....	9
2.2.1 Lập trình client và server với TCP socket.....	9
2.2.2 Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi.....	12
TÀI LIỆU THAM KHẢO	20

DANH MỤC CÁC HÌNH VẼ

Hình 1 – Socket	5
Hình 2 – Chức năng của socket.....	6
Hình 3 – Lập trình socket – 1	8
Hình 4 – Lập trình socket – 2.....	8
Hình 5 – Lập trình và chạy Server	9
Hình 6 – Lập trình và chạy Client để phản hồi lại Server	10
Hình 7 – Phía Server sau khi nhận phản hồi từ Client	10
Hình 8 – Khởi động Wireshark và chọn Adapter for loopback traffic capture.....	11
Hình 9 – Bắt thông tin được gửi từ Server đến Client	11
Hình 10 – Bắt thông tin được gửi từ Client đến Server	12
Hình 11 – Sửa đổi chương trình bên Server	13
Hình 12 – Sửa đổi chương trình bên Client	13
Hình 13 – Kết quả trả về bên Server	14
Hình 14 – Kết quả khi chạy chương trình bên Client	14
Hình 15 – Client gửi thông điệp bản rõ đến Server.....	15
Hình 16 – Client gửi thông điệp + key đã được mã hóa SHA512	15
Hình 17 – Server gửi phản hồi đến Client sau khi nhận thông điệp + key không thay đổi	16
Hình 18 – Thay đổi key ở Client → Thông điệp mã hash thay đổi	16
Hình 19 – Bên Server giữ nguyên.....	17
Hình 20 – Bên Server: mã hash của thông điệp đã bị thay đổi → Không toàn vẹn dữ liệu	17
Hình 21 – Kết quả, bên Client nhận được thông báo rằng dữ liệu đã bị mất mát.....	18
Hình 22 – Thông điệp gửi từ Client đến Server	18
Hình 23 – Khi truyền từ Client → Server, mã hash đã bị thay đổi	19
Hình 24 – Server phản hồi Client: Dữ liệu đã bị mất mát.....	19

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh/Giải thích	Thuật ngữ tiếng Việt/Giải thích
TCP	Transmission Control Protocol	Giao thức điều khiển truyền vận
UDP	User Datagram Protocol	Giao thức dữ liệu người dùng
IP	Internet Protocol	Giao thức Internet
SHA512	Secure Hash Algorithm 512-bit	Thuật toán băm an toàn với độ dài 512 bit

CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH

1.1 Mục đích

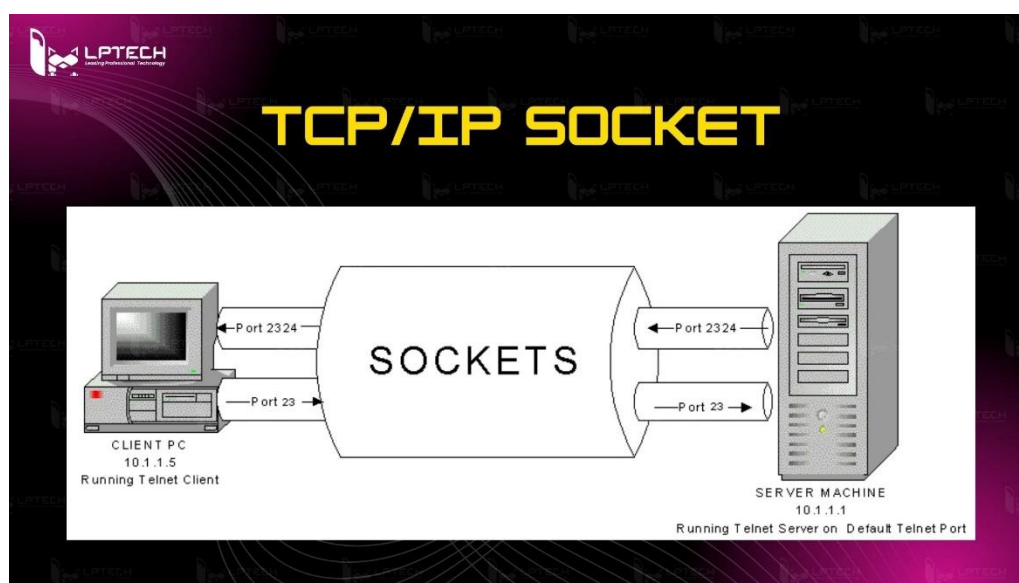
Sinh viên hiểu về cơ chế client/server và có thể tự lập trình client/server dựa trên socket, sau đó thực hiện ca đặt giao thức đơn giản để trao đổi thông tin an toàn.

1.2 Tìm hiểu lý thuyết

Tìm hiểu về các khái niệm liên quan đến lập trình socket với TCP

1.2.1 Socket là gì?

Socket là điểm cuối end-point trong liên kết truyền thông hai chiều (two-way communication) biểu diễn kết nối giữa Client – Server. Các lớp Socket được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới.



Hình 1 – Socket

Socket là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều, hay còn gọi là two-way communication để kết nối 2 process trò chuyện với nhau. Điểm cuối (endpoint) của liên kết này được gọi là socket.

Một chức năng khác của socket là giúp các tầng *TCP* hoặc *TCP Layer* định danh ứng dụng mà dữ liệu sẽ được gửi tới thông qua sự ràng buộc với một cổng port (thể hiện là một con số cụ thể), từ đó tiến hành kết nối giữa client và server.

1.2.2 Tại sao người dùng lại cần dùng đến socket?

Người dùng cần đến socket vì nó là một cách tiêu biểu để thiết lập và quản lý kết nối mạng giữa các thiết bị và ứng dụng trên Internet. Dưới đây là một số lý do chính:

Giao tiếp mạng: Socket cho phép ứng dụng gửi và nhận dữ liệu qua mạng, cho phép giao tiếp giữa các thiết bị từ xa. Điều này là cần thiết trong nhiều ứng dụng như trò chơi trực tuyến, chat, truyền tệp, và nhiều ứng dụng mạng khác.

Tính linh hoạt: Socket hỗ trợ nhiều loại kết nối mạng khác nhau như TCP, UDP, và các giao thức khác, cung cấp sự linh hoạt trong cách ứng dụng giao tiếp với nhau.

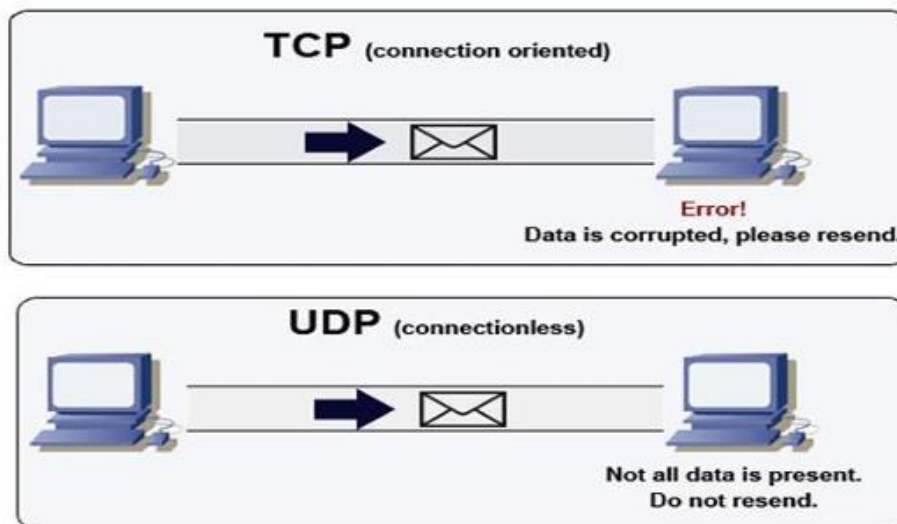
Phân phối dữ liệu: Sử dụng socket, dữ liệu có thể được phân phối từ một nguồn tới nhiều đích hoặc ngược lại. Điều này cho phép triển khai các ứng dụng mạng phức tạp như streaming video, trò chơi trực tuyến, và các ứng dụng đa người dùng.

Tương thích đa nền tảng: Socket có sẵn trên nhiều nền tảng và hệ điều hành khác nhau, từ máy tính cá nhân đến thiết bị di động, giúp cho việc phát triển ứng dụng mạng trở nên dễ dàng và linh hoạt hơn.

Kiểm soát độ trễ: Sử dụng socket, người dùng có thể kiểm soát và tối ưu hóa độ trễ trong việc truyền và nhận dữ liệu qua mạng, điều này quan trọng đặc biệt trong các ứng dụng đòi hỏi độ trễ thấp như trò chơi trực tuyến và ứng dụng thời gian thực.

Tóm lại, socket là một công cụ quan trọng trong việc phát triển các ứng dụng mạng, cho phép giao tiếp hiệu quả và đa dạng giữa các thiết bị và ứng dụng trên Internet.

1.2.3 Điều kiện để socket hoạt động?



Hình 2 – Chức năng của socket

Như đã đề cập trước đó, chức năng của socket là kết nối giữa client và server thông qua TCP/IP và UDP để truyền và nhận dữ liệu qua Internet. Giao diện lập trình ứng dụng mạng này chỉ có thể hoạt động khi đã có thông tin về thông số IP và số hiệu cổng của 2 ứng dụng cần trao đổi dữ liệu cho nhau.

Hai ứng dụng cần truyền thông tin phải đáp ứng điều kiện sau thì socket mới có thể hoạt động:

- Hai ứng dụng có thể nằm cùng trên một máy hoặc 2 máy khác nhau
- Trong trường hợp 2 ứng dụng cùng nằm trên một máy, số hiệu cổng không được trùng nhau.

1.2.4 Phân loại socket

Socket có thể được phân loại chủ yếu dựa trên hai tiêu chí: giao thức và cách sử dụng. Dưới đây là một số phân loại phổ biến của socket:

- *Theo giao thức:*
 - **Socket TCP (Transmission Control Protocol):** Sử dụng giao thức TCP để thiết lập kết nối đáng tin cậy, có kiểm soát lỗi và bảo đảm dữ liệu đến được đích một cách chính xác.
 - **Socket UDP (User Datagram Protocol):** Sử dụng giao thức UDP, không đảm bảo tính toàn vẹn hoặc độ tin cậy, nhưng thích hợp cho các ứng dụng cần truyền dữ liệu nhanh mà không cần quá nhiều kiểm soát.
- *Theo cách sử dụng:*
 - **Socket Server:** Là socket được sử dụng để lắng nghe và chấp nhận kết nối từ các client khác.
 - **Socket Client:** Là socket được sử dụng để thiết lập kết nối và truyền dữ liệu đến một socket server.

Mỗi loại socket có mục đích và ứng dụng khác nhau trong lập trình mạng. Sử dụng đúng loại socket sẽ giúp đảm bảo tính ổn định và hiệu suất của ứng dụng mạng.

1.2.5 Lập trình socket với TCP/IP

Các tiến trình mà muốn truyền thông với nhau thì sẽ gửi thông điệp thông qua các socket. Socket là cánh cửa của tiến trình ứng dụng và giao thức tầng transport (ở đây là TCP).

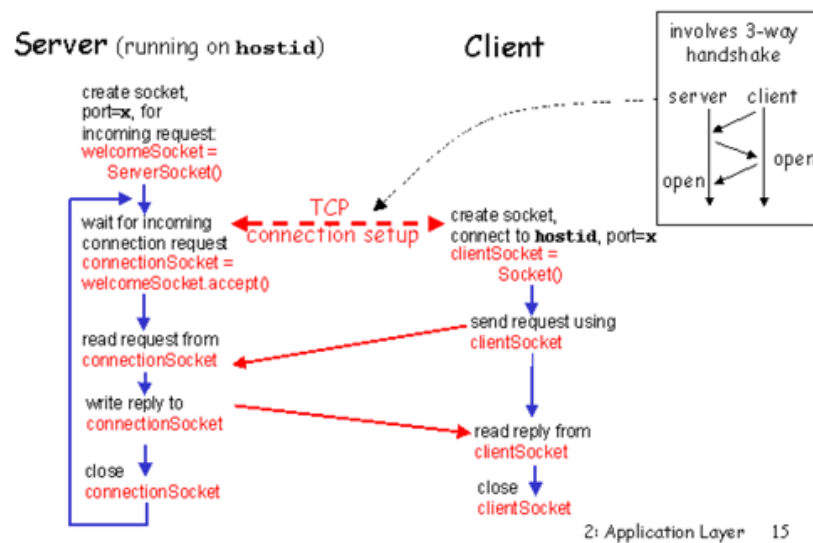
Để có thể tương tác với nhau và máy chủ có thể nhận liên lạc từ máy khách thì máy chủ phải luôn sẵn sàng. Điều này có 2 nghĩa, thứ nhất, giống như trường hợp của UDP, một tiến trình của máy chủ phải được chạy trước khi máy khách khởi tạo liên lạc đến. Thứ hai, chương trình chủ phải tạo ra 1 socket để sẵn sàng chấp nhận kết nối từ tiến trình khách.

Khi tiến trình chủ đã chạy, lúc này tiến trình khách sẽ tạo ra 1 socket TCP để có thể kết nối đến máy chủ. Trong khi máy khách đang tạo TCP socket, nó sẽ đặc tả địa chỉ IP, số cổng của tiến trình chủ.

Khi socket của tiến trình khách vừa được tạo, TCP trên máy khách sẽ tiến hành thực hiện quá trình bắt tay 3 bước và thiết lập kết nối TCP tới máy chủ.

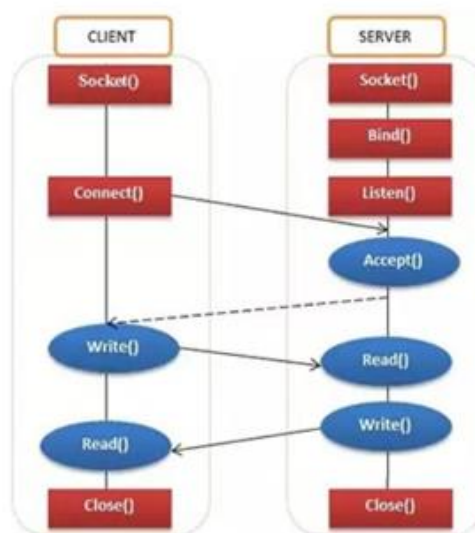
Trong quá trình bắt tay 3 bước, khi tiến trình chủ nhận thấy tiến trình khách, nó sẽ tự tạo ra 1 socket mới chỉ dành riêng cho tiến trình khách đó. Khi được máy khách gõ cửa, chương trình kích hoạt với phương thức `accept()`. Cuối quá trình bắt tay 3 bước, một kết nối TCP tồn tại giữa socket của máy khách và socket của máy chủ.

Client/server socket interaction: TCP



Hình 3 – Lập trình socket – 1

Mô tả quá trình:



Hình 4 – Lập trình socket – 2

- Trước tiên chúng ta sẽ tạo ra một máy chủ bằng cách mở một socket - `Socket()`
- Sau đó chúng ta sẽ liên kết nó với một host hoặc một máy và một port - `Bind()`
- Tiếp theo server sẽ bắt đầu lắng nghe trên port đó - `Listen()`
- Yêu cầu kết nối từ client được gửi tới server - `Connect()`
- Server sẽ accept yêu cầu từ client và sau đó kết nối được thiết lập - `Accept()`
- Bây giờ cả hai đều có thể gửi và nhận tin tại thời điểm đó - `Read()` / `Write()`
- Và cuối cùng khi hoàn thành chúng có thể đóng kết nối - `Close()`

CHƯƠNG 2. NỘI DUNG THỰC HÀNH

2.1 Chuẩn bị môi trường

- Môi trường Python hoặc Java để chạy được ứng dụng client/server đã lập trình.
- Phần mềm Wireshark.

2.2 Các bước thực hiện

2.2.1 Lập trình client và server với TCP socket

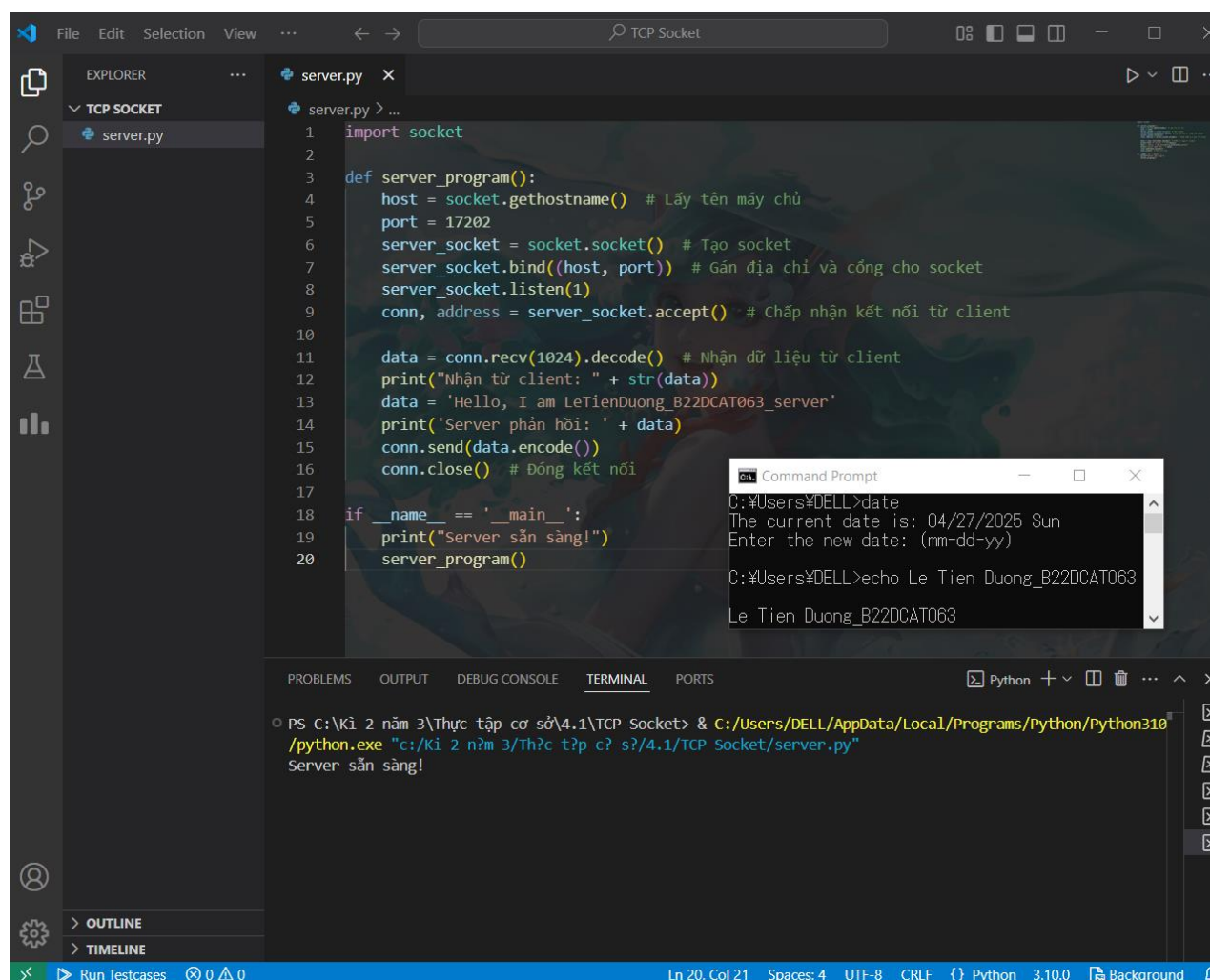
Lập trình client.

Lập trình server.

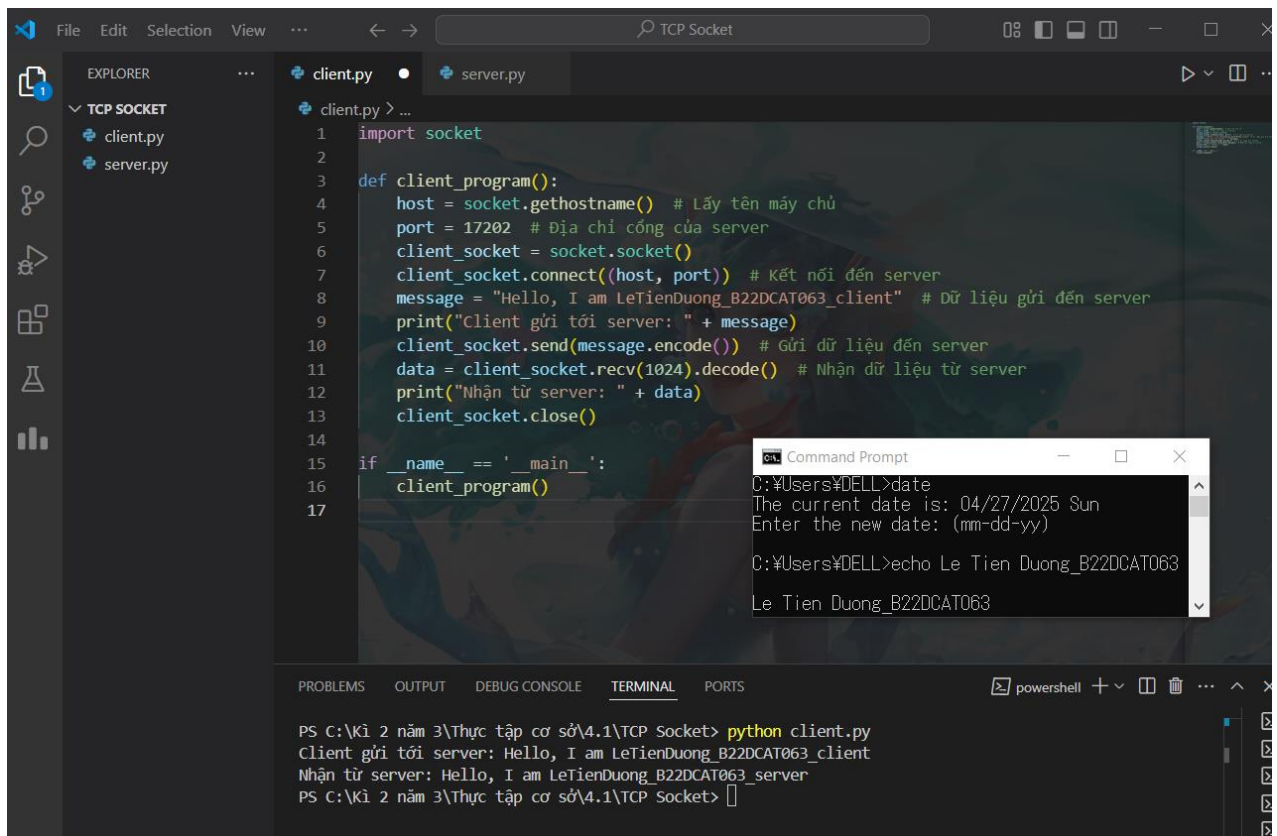
Chạy server sau đó chạy client.

Client gửi thông điệp cá nhân hóa cho server: “Hello, I am LeTienDuong_B22DCAT063_client”.

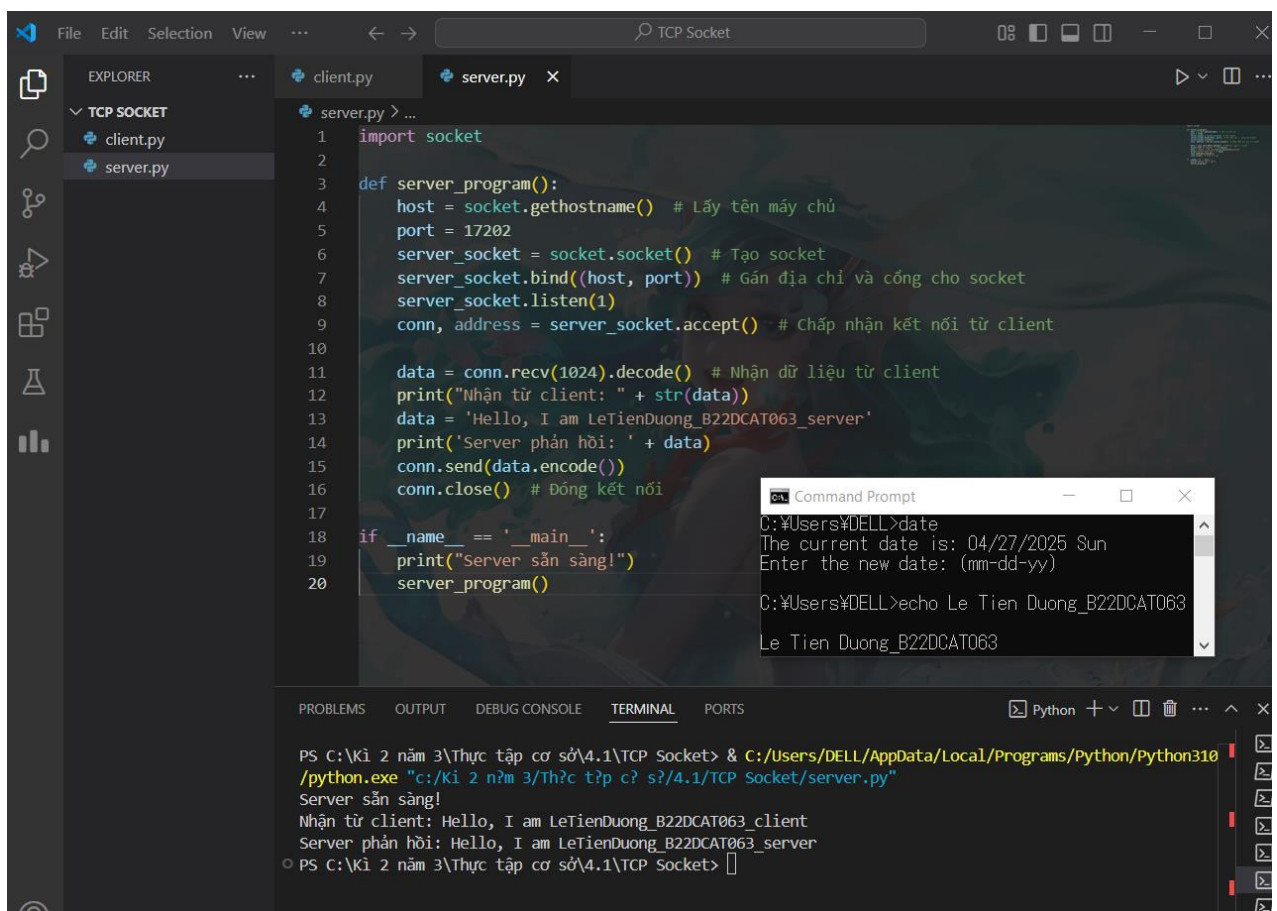
Server nhận được hiển thị thông điệp nhận được và gửi lại client thông điệp: Server gửi lại “Hello, I am LeTienDuong_B22DCAT063_server”.



Hình 5 – Lập trình và chạy Server

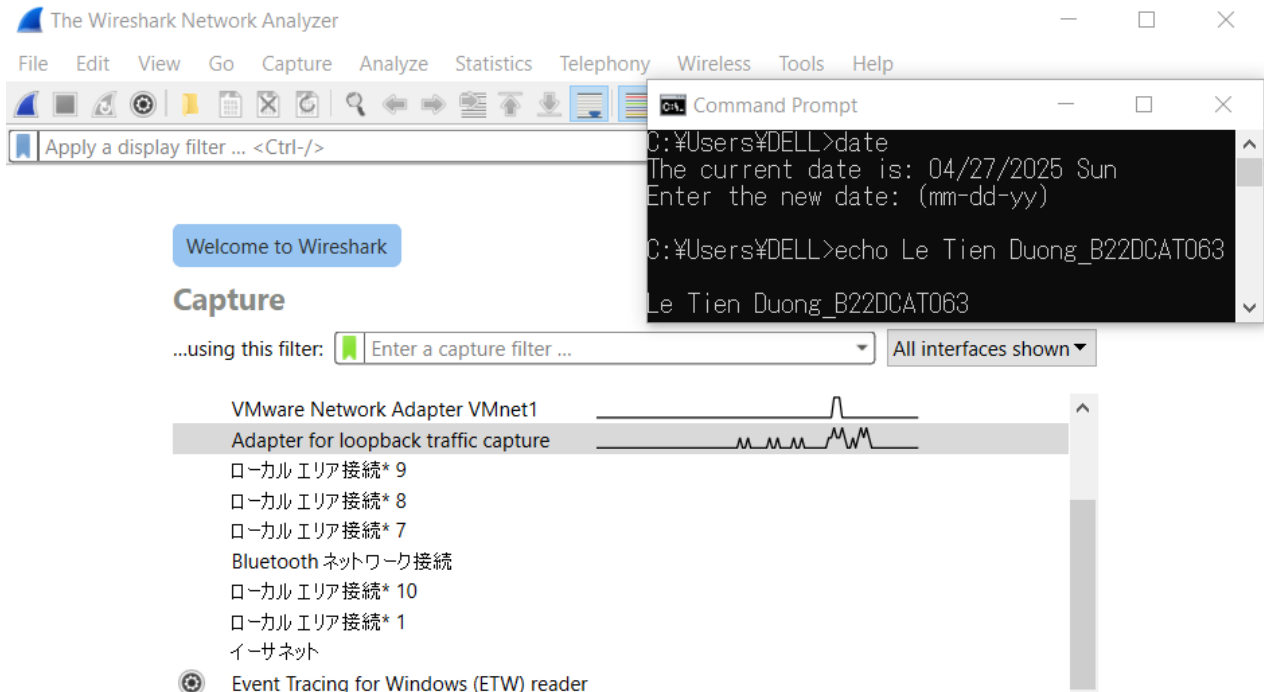


Hình 6 – Lập trình và chạy Client để phản hồi lại Server



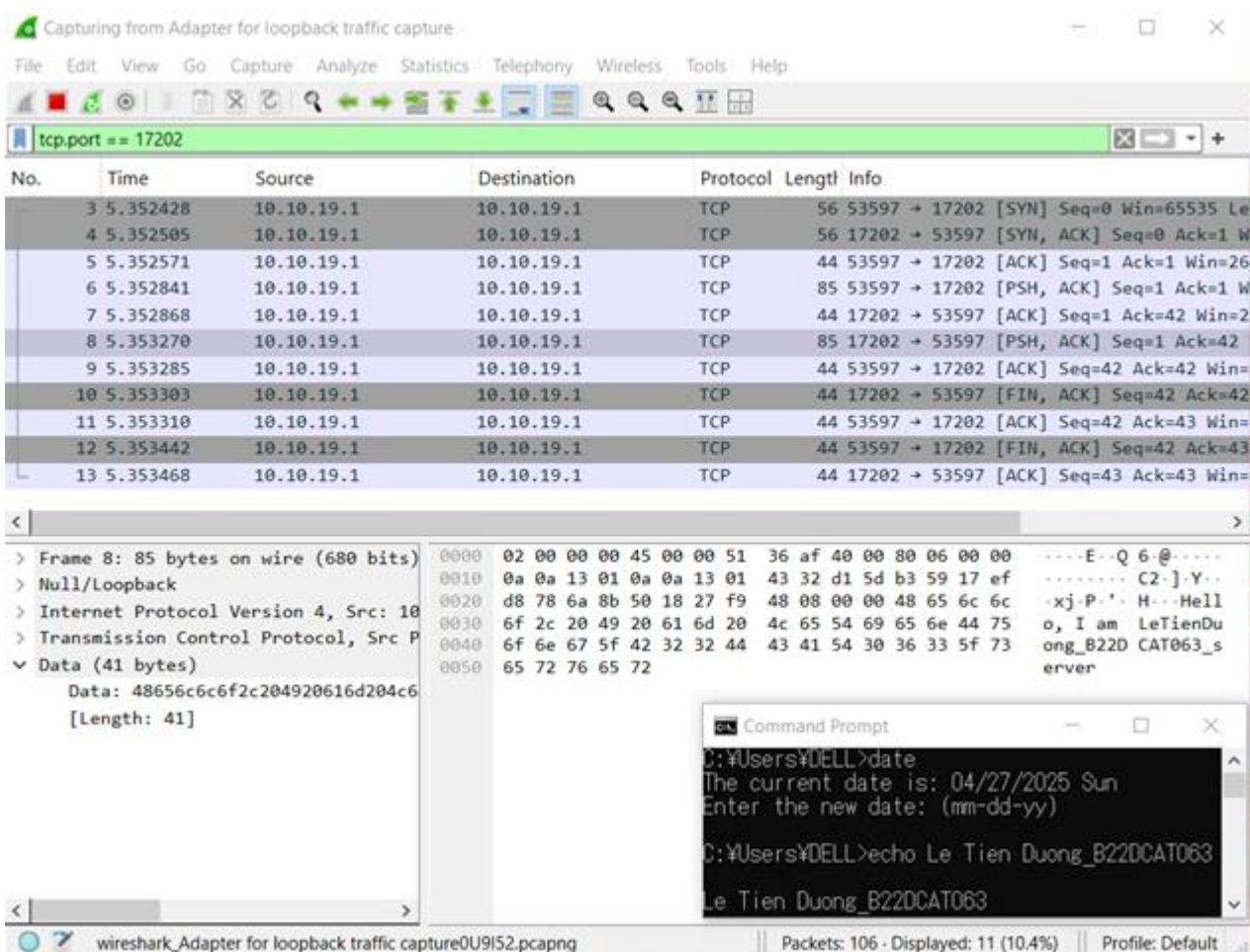
Hình 7 – Phía Server sau khi nhận phản hồi từ Client

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

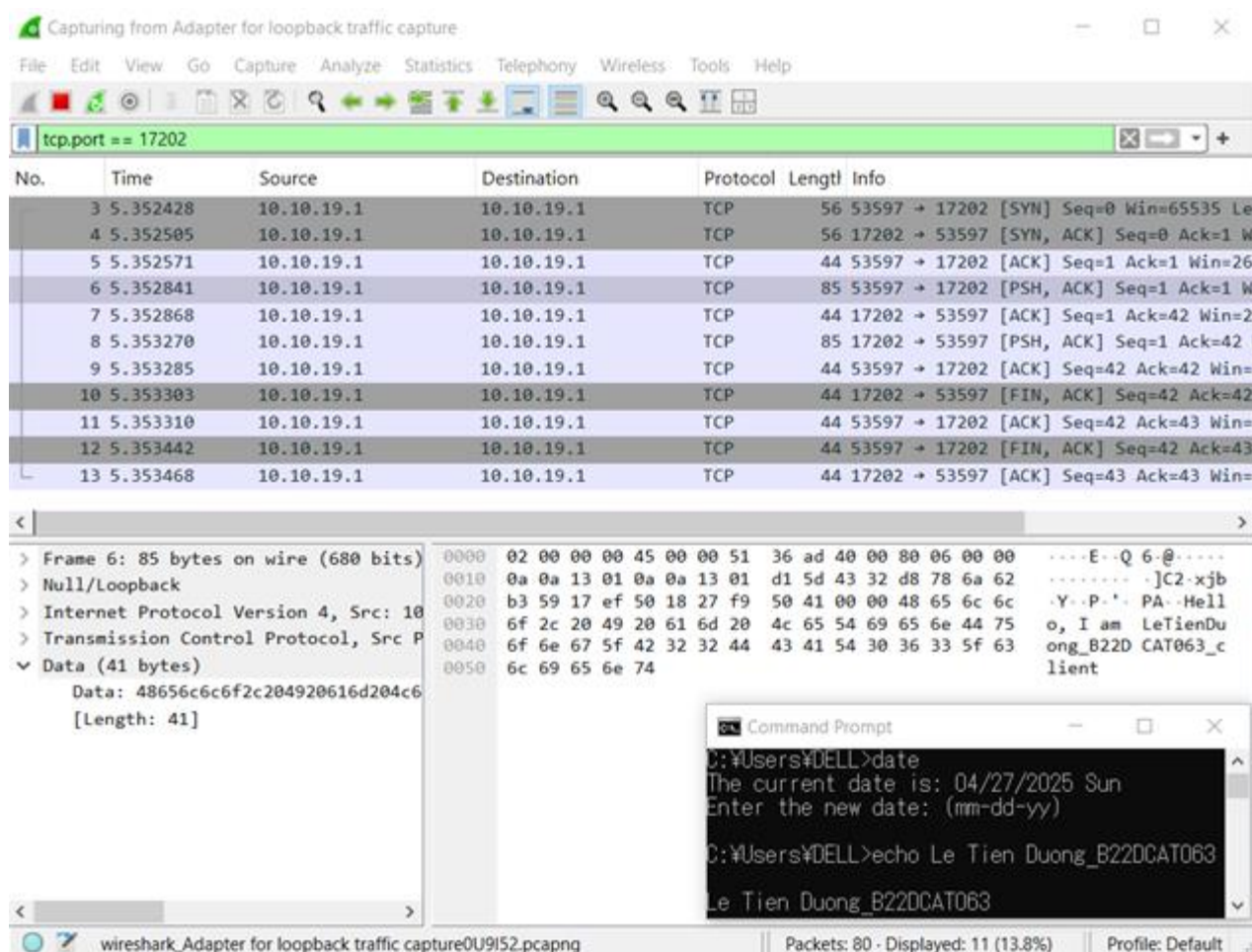


Hình 8 – Khởi động Wireshark và chọn Adapter for loopback traffic capture

Tiến hành kết nối lại Server với Client và bắt gói tin.



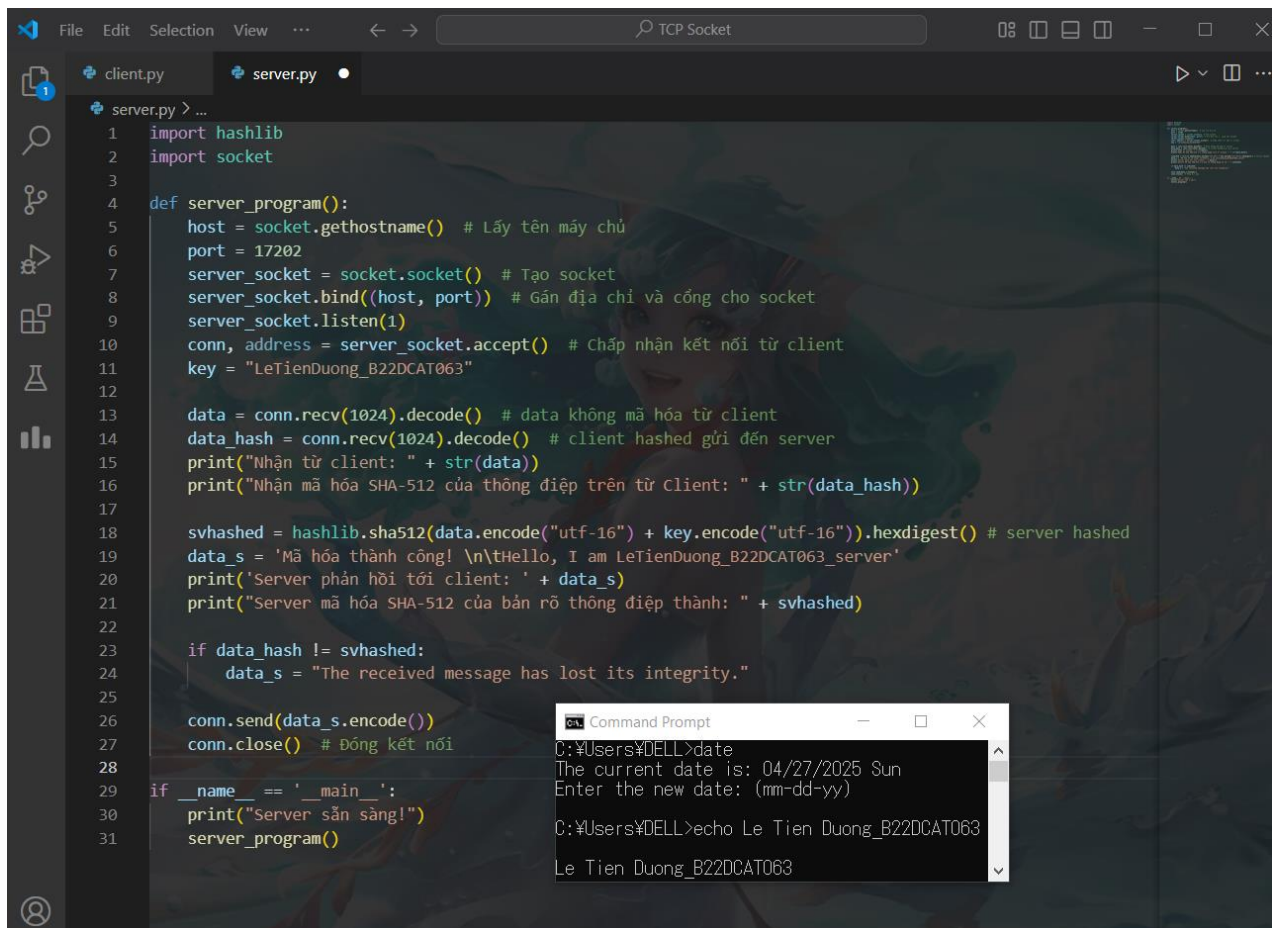
Hình 9 – Bắt thông tin được gửi từ Server đến Client



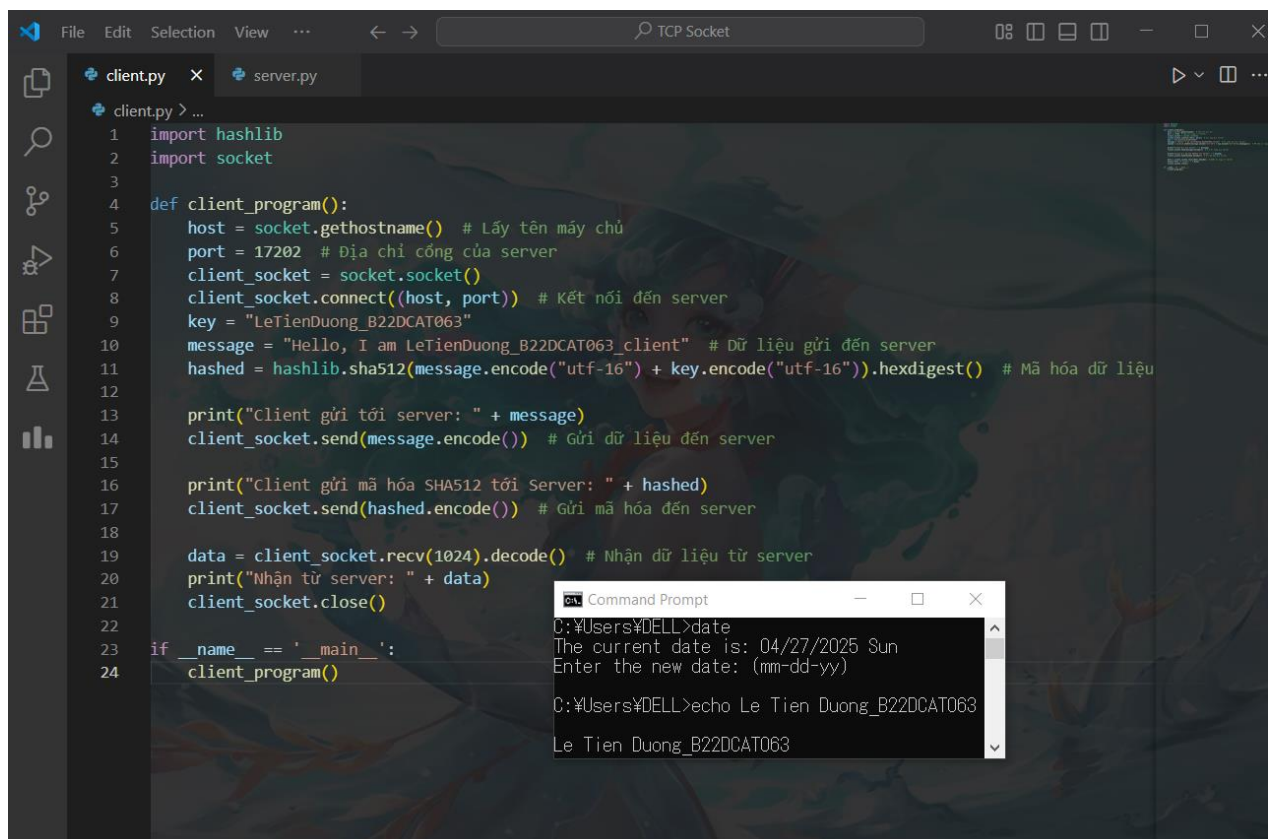
Hình 10 – Bắt thông tin được gửi từ Client đến Server

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.

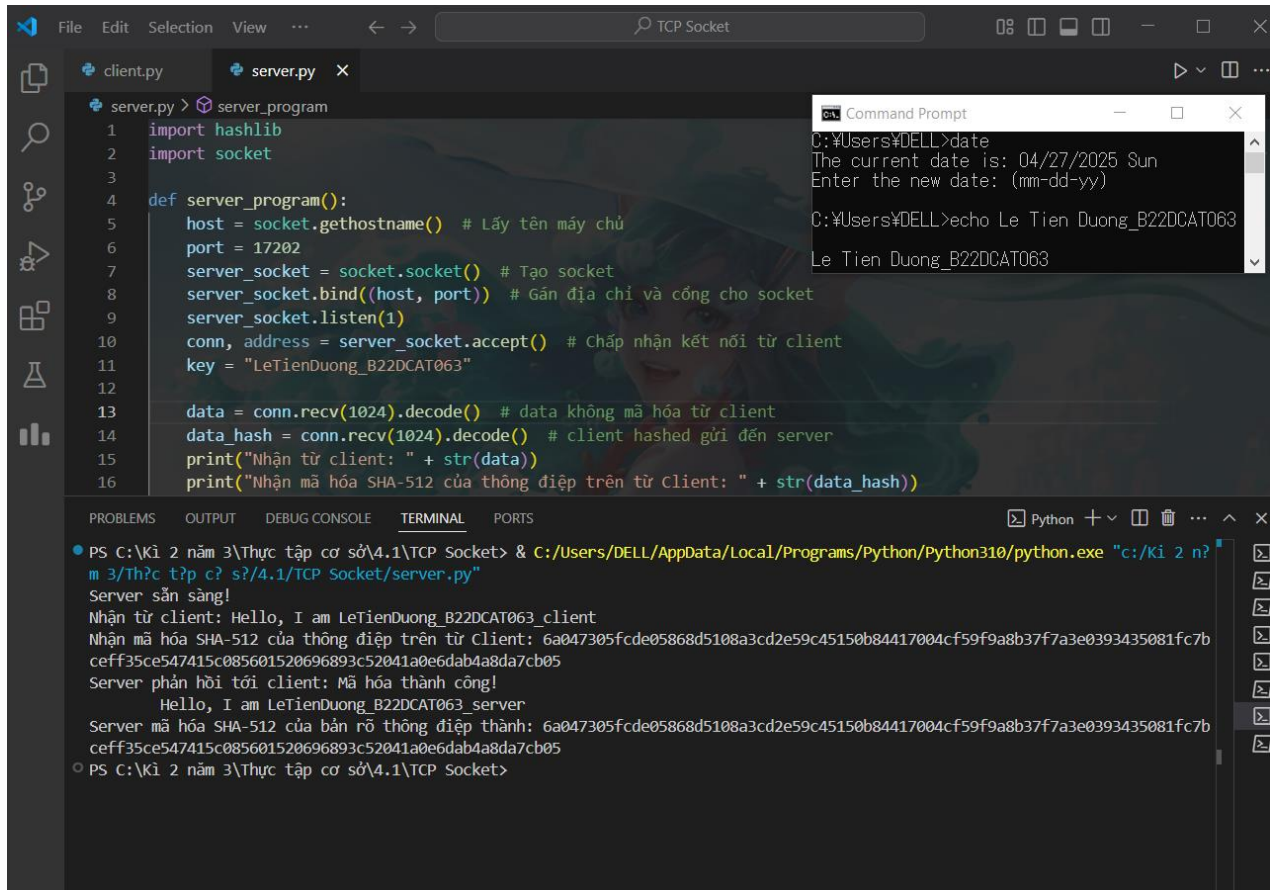


Hình 11 – Sửa đổi chương trình bên Server



Hình 12 – Sửa đổi chương trình bên Client

Tiến hành chạy lần lượt chương trình bên Server và bên Client để xem kết quả trả về

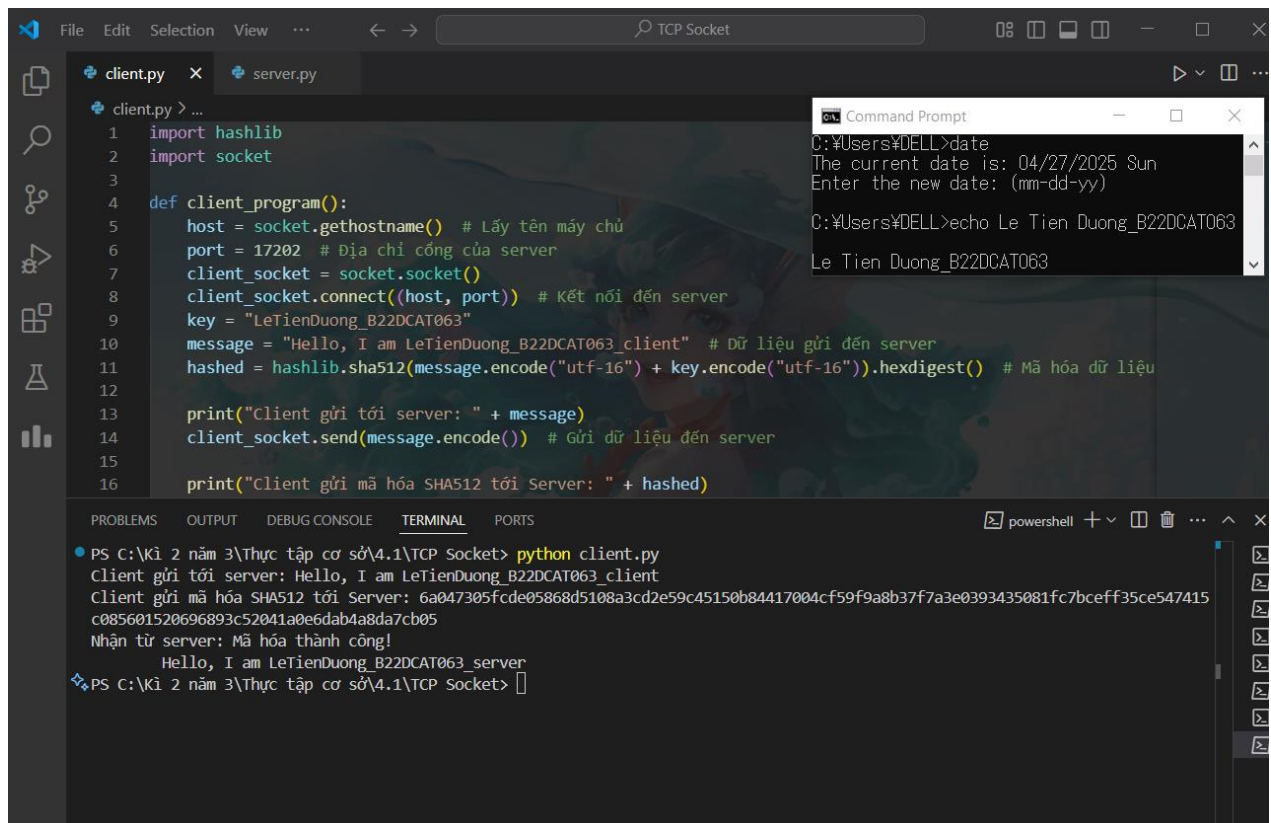


```
server.py
1 import hashlib
2 import socket
3
4 def server_program():
5     host = socket.gethostname() # Lấy tên máy chủ
6     port = 17202
7     server_socket = socket.socket() # Tạo socket
8     server_socket.bind((host, port)) # Gán địa chỉ và cổng cho socket
9     server_socket.listen(1)
10    conn, address = server_socket.accept() # Chấp nhận kết nối từ client
11    key = "LeTienDuong_B22DCAT063"
12
13    data = conn.recv(1024).decode() # data không mã hóa từ client
14    data_hash = conn.recv(1024).decode() # client hashed gửi đến server
15    print("Nhận từ client: " + str(data))
16    print("Nhận mã hóa SHA-512 của thông điệp trên từ Client: " + str(data_hash))
```

```
Command Prompt
C:\Users\DELL>date
The current date is: 04/27/2025 Sun
Enter the new date: (mm-dd-yy)
C:\Users\DELL>echo Le Tien Duong_B22DCAT063
Le Tien Duong_B22DCAT063
```

```
PS C:\Ki 2 năm 3\Thực tập cơ sở\4.1\TCP Socket> & C:/Users/DELL/AppData/Local/Programs/Python/Python310/python.exe "c:/ki 2 n?
m 3/Th?c t?p c? s?/4.1/TCP Socket/server.py"
Server sẵn sàng!
Nhận từ client: Hello, I am LeTienDuong_B22DCAT063_client
Nhận mã hóa SHA-512 của thông điệp trên từ Client: 6a047305fcde05868d5108a3cd2e59c45150b84417004cf59f9a8b37f7a3e0393435081fc7bceff35ce547415c085601520696893c52041a0e6dab4a8da7cb05
Server phản hồi tới client: Mã hóa thành công!
Hello, I am LeTienDuong_B22DCAT063_server
Server mã hóa SHA-512 của bản rõ thông điệp thành: 6a047305fcde05868d5108a3cd2e59c45150b84417004cf59f9a8b37f7a3e0393435081fc7bceff35ce547415c085601520696893c52041a0e6dab4a8da7cb05
PS C:\Ki 2 năm 3\Thực tập cơ sở\4.1\TCP Socket>
```

Hình 13 – Kết quả trả về bên Server



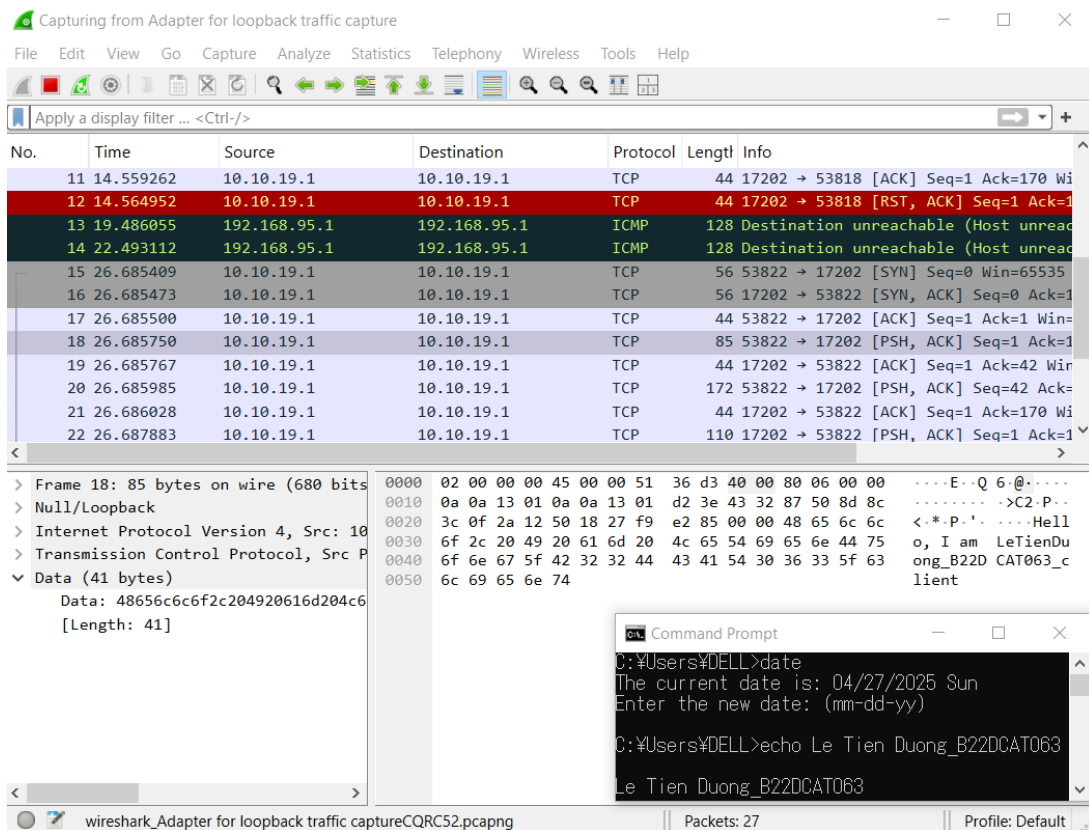
```
client.py
1 import hashlib
2 import socket
3
4 def client_program():
5     host = socket.gethostname() # Lấy tên máy chủ
6     port = 17202 # Địa chỉ cổng của server
7     client_socket = socket.socket()
8     client_socket.connect((host, port)) # Kết nối đến server
9     key = "LeTienDuong_B22DCAT063"
10    message = "Hello, I am LeTienDuong_B22DCAT063_client" # Dữ liệu gửi đến server
11    hashed = hashlib.sha512(message.encode("utf-16") + key.encode("utf-16")).hexdigest() # Mã hóa dữ liệu
12
13    print("Client gửi tới server: " + message)
14    client_socket.send(message.encode()) # Gửi dữ liệu đến server
15
16    print("Client gửi mã hóa SHA512 tới Server: " + hashed)
```

```
Command Prompt
C:\Users\DELL>date
The current date is: 04/27/2025 Sun
Enter the new date: (mm-dd-yy)
C:\Users\DELL>echo Le Tien Duong_B22DCAT063
Le Tien Duong_B22DCAT063
```

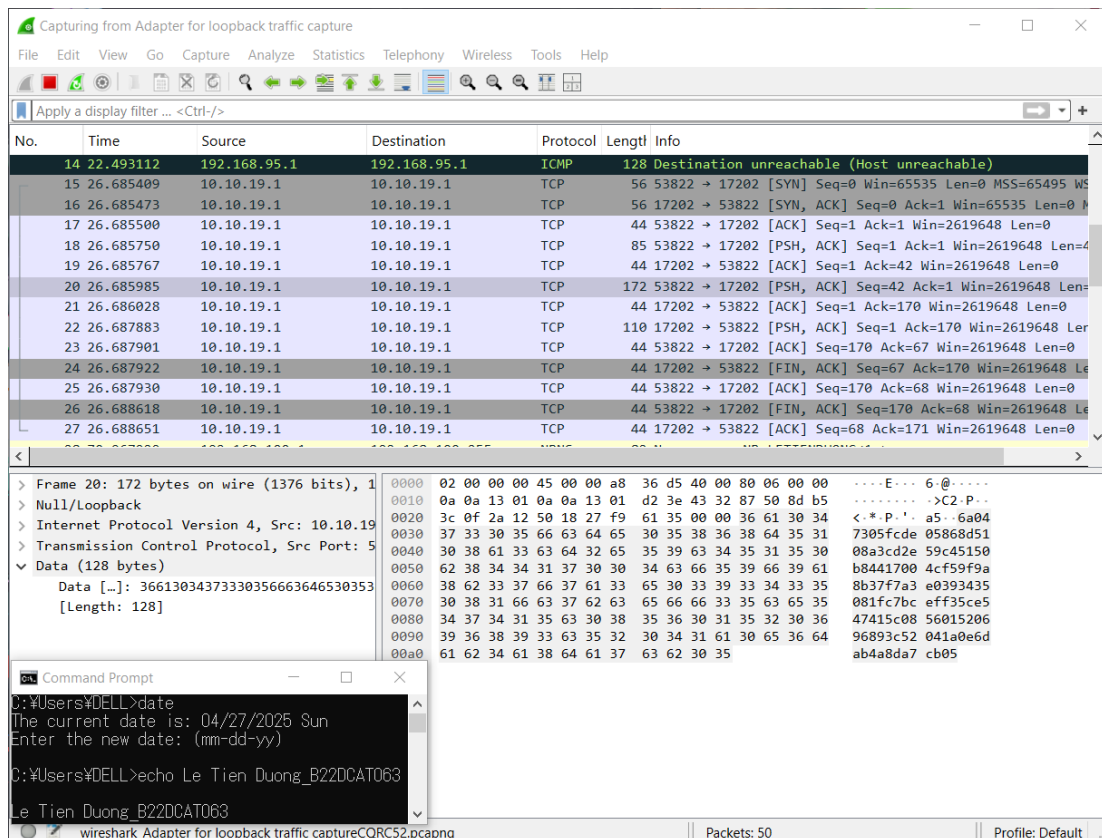
```
PS C:\Ki 2 năm 3\Thực tập cơ sở\4.1\TCP Socket> python client.py
Client gửi tới server: Hello, I am LeTienDuong_B22DCAT063_client
Client gửi mã hóa SHA512 tới Server: 6a047305fcde05868d5108a3cd2e59c45150b84417004cf59f9a8b37f7a3e0393435081fc7bceff35ce547415c085601520696893c52041a0e6dab4a8da7cb05
Nhận từ server: Mã hóa thành công!
Hello, I am LeTienDuong_B22DCAT063_server
PS C:\Ki 2 năm 3\Thực tập cơ sở\4.1\TCP Socket>
```

Hình 14 – Kết quả khi chạy chương trình bên Client

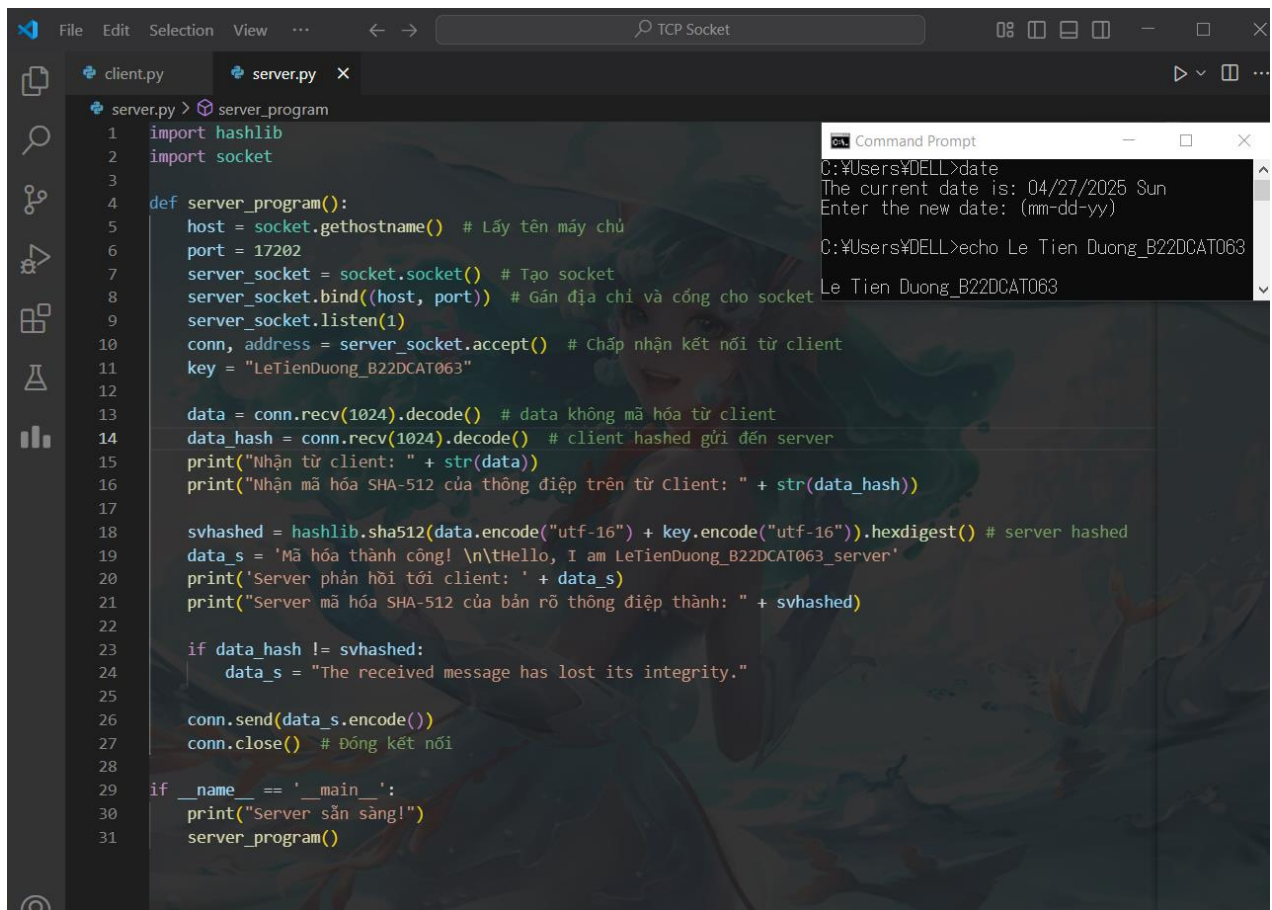
Bắt các bản tin trao đổi giữa Client và Server trong Wireshark trong trường hợp giống key.



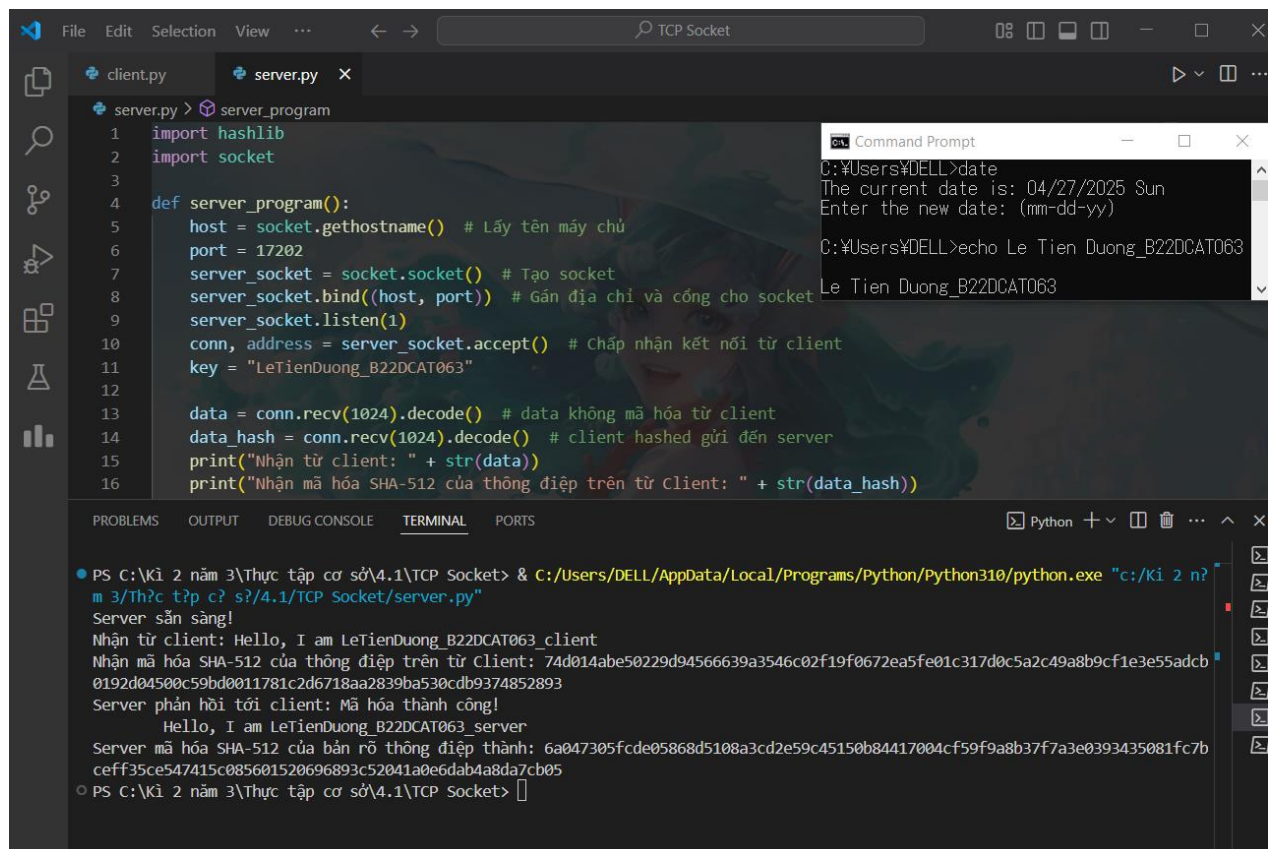
Hình 15 – Client gửi thông điệp bản rõ đến Server



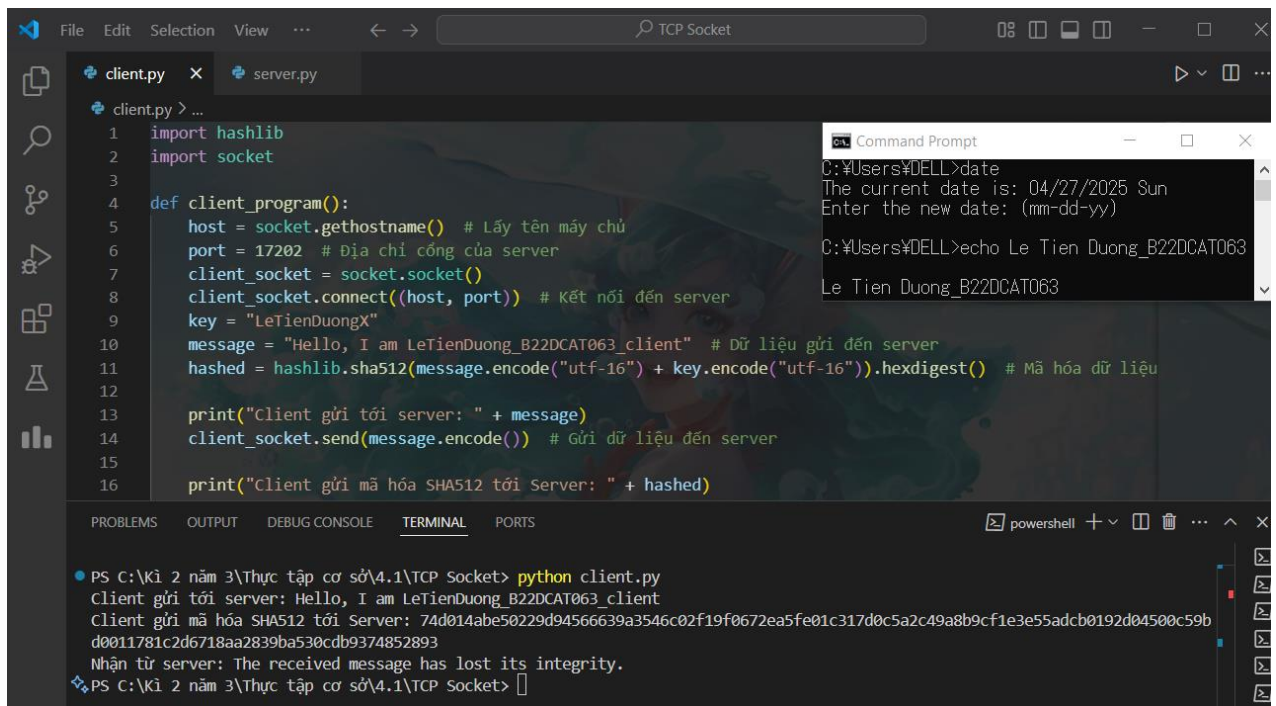
Hình 16 – Client gửi thông điệp + key đã được mã hóa SHA512



Hình 19 – Bên Server giữ nguyên

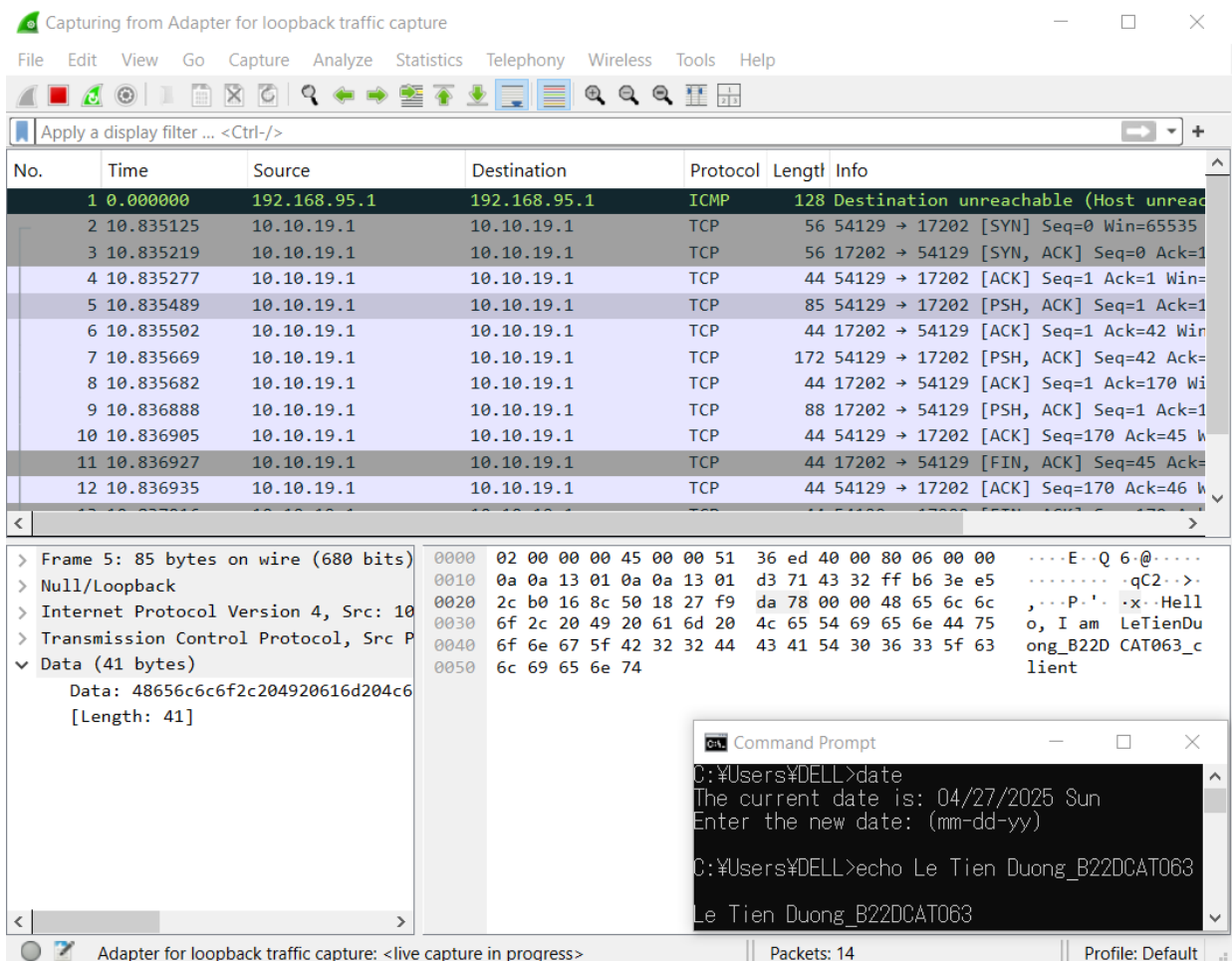


Hình 20 – Bên Server: mã hash của thông điệp đã bị thay đổi → Không toàn vẹn dữ liệu



Hình 21 – Kết quả, bên Client nhận được thông báo rằng dữ liệu đã bị mất mát

Bắt được các bản tin trao đổi giữa client và server trong Wireshark trong trường hợp key bị thay đổi.



Hình 22 – Thông điệp gửi từ Client đến Server

Adapter for loopback traffic capture: <live capture in progress>

Hình 23 – Khi truyền từ Client → Server, mã hash đã bị thay đổi

Adapter for loopback traffic capture: <live capture in progress> Packets: 52 Profile: Default

Hình 24 – Server phản hồi Client: Dữ liệu đã bị mất mát

TÀI LIỆU THAM KHẢO

- [1] Tham khảo tài liệu: Chapter 2: Application Layer V8.1 (9/2020) tại địa chỉ http://gaia.cs.umass.edu/kurose_ross/ppt.php (chú ý ví dụ từ trang 105).
- [2] Hướng dẫn code Socket, xem ở đây