

**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:

B22DCAT034 Trương Quốc Bảo

Giảng viên hướng dẫn: TS. Đinh Trường Duy

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 Cơ chế client/server.....	5
1.2.2 Lập trình socket với TCP	7
CHƯƠNG 2. NỘI DUNG THỰC HÀNH	10
2.1 Chuẩn bị môi trường	10
2.2 Các bước thực hiện.....	10
2.2.1 Lập trình client và server với TCP socket.....	10
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	19

DANH MỤC CÁC HÌNH VẼ

Hình 1 Cách thức hoạt động của client – server.....	5
Hình 2 Mô hình OSI và TCP/IP	7
Hình 3 Ví dụ cơ bản	9
Hình 4 Lập trình cho Client.....	10
Hình 5 Lập trình cho Server	10
Hình 6 Khởi chạy Server.....	11
Hình 7 Khởi chạy Client	11
Hình 8 Khởi động Wireshark để bắt các gói tin.....	11
Hình 9 Wireshark bắt được thông điệp 1	12
Hình 10 Wireshark bắt được thông điệp 2	12
Hình 11 Lập trình Server với key.....	13
Hình 12 Lập trình Client với key	14
Hình 13 Khởi chạy Server.....	14
Hình 14 Khởi chạy Client	15
Hình 15 Wireshark bắt được thông điệp đã mã hóa.....	15
Hình 16 Thay đổi giá trị key của Client	16
Hình 17 Client truyền thông điệp.....	16
Hình 18 Thông điệp đến server đã bị khác đi	17
Hình 19 Wireshark bắt được mã hash của thông điệp	17
Hình 20 Thông báo thông điệp đã mất mát.....	18

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 tải
UDP	User Datagram Protocol	Giao thức dữ liệu người dùng
OSI	Open Systems Interconnection	Mô hình tham chiếu kết nối các hệ thống mở
IP	Internet Protocol	Giao thức Internet
WWW	World Wide Web	Mạng lưới toàn cầu
FTP	File Transfer Protocol	Giao thức truyền tệp

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

1.2.1 Cơ chế client/server

Mô hình client-server là một mô hình mạng máy tính gồm hai thành phần chính là client và server. Client sẽ là bên yêu cầu dịch vụ cài đặt cũng như lưu trữ tài nguyên từ server. Khi client gửi yêu cầu dữ liệu đến server qua Internet, server sẽ xử lý yêu cầu và gửi các gói dữ liệu cho client.

Mô hình Client Server sử dụng cấu trúc phân tán để phân chia nhiệm vụ giữa client và server. Mô hình còn có một số ứng dụng phổ biến như email và World Wide Web (WWW)...

Client

Trong mô hình Client Server, Client là các máy tính truy cập để sử dụng dịch vụ (còn gọi là Host) và có khả năng nhận thông tin cụ thể từ nhà cung cấp dịch vụ (Server).

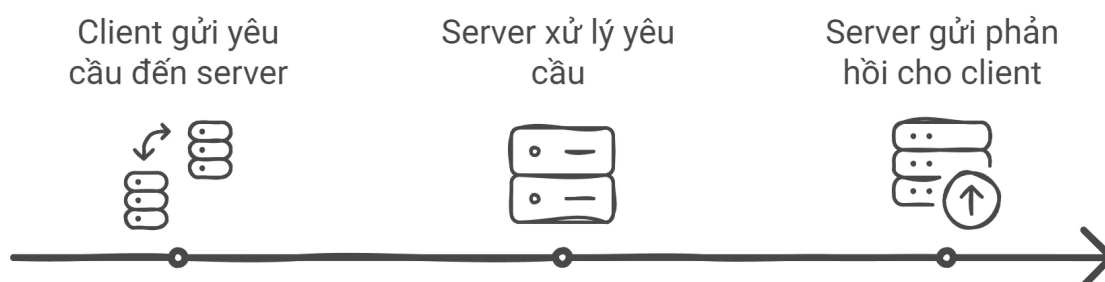
Server

Server là một máy chủ hay phương tiện để cung cấp các dịch vụ đến client. Thường cấu hình của server rất mạnh và luôn kết nối mạng để có thể phục vụ client 24/7.

Cách thức hoạt động cụ thể của mô hình client – server:

- Đầu tiên, client sẽ gửi câu đến server thông qua một giao thức mạng.
- Tiếp theo, server sẽ xử lý các yêu cầu từ client
- Cuối cùng, server sẽ trả kết quả về cho client thường là một trang web, email, file hoặc thông báo lỗi.

HOẠT ĐỘNG CỦA MÔ HÌNH CLIENT - SERVER



Hình 1 Cách thức hoạt động của client – server

Ví dụ mô hình Client – Server

- Email server: Các thư điện tử được gửi từ client sẽ được server tiếp nhận email, lưu trữ và gửi email đó đến địa chỉ nhận.
- File server: Khi client chia sẻ thông tin đến file server sẽ được lưu trữ và truyền file đi. Ngoài ra, người dùng có thể upload hoặc download các file lên server thông qua giao thức FTP/web browser.
- World wide web (WWW): Thông qua trình duyệt web người dùng có thể truy cập vào các trang web đã được lưu trữ trên server.
- Giao dịch online: Các giao dịch giữa người mua và bán như mua bán hàng hóa, dịch vụ đều được thực hiện thông qua server.
- Game online: Các game trực tuyến sẽ được lưu trữ trên server và người chơi sẽ truy cập vào để chơi.

Ưu điểm:

- Khả năng tập trung: Client server có tích hợp Centralization (Tập trung hóa) giúp quản trị viên mạng dễ dàng quản lý và giải quyết vấn đề.
- Tính bảo mật: Dữ liệu được bảo vệ tốt hơn và tránh truy cập trái phép nhờ vào cấu trúc tập trung và các biện pháp hiện đại như firewall, mã hóa...
- Khả năng mở rộng: Người dùng có thể tăng số lượng tài nguyên như số lượng client hoặc server mà không bị gián đoạn nhờ khả năng mở rộng tốt của mô hình.
- Khả năng truy cập: Mô hình client-server không phân biệt mà cho phép người dùng truy cập thông tin từ bất kỳ vị trí hoặc nền tảng nào.
- Hiệu quả cao: Server có thể xử lý nhiều yêu cầu cùng lúc, giúp tăng hiệu suất tài nguyên cho client.

Nhược điểm:

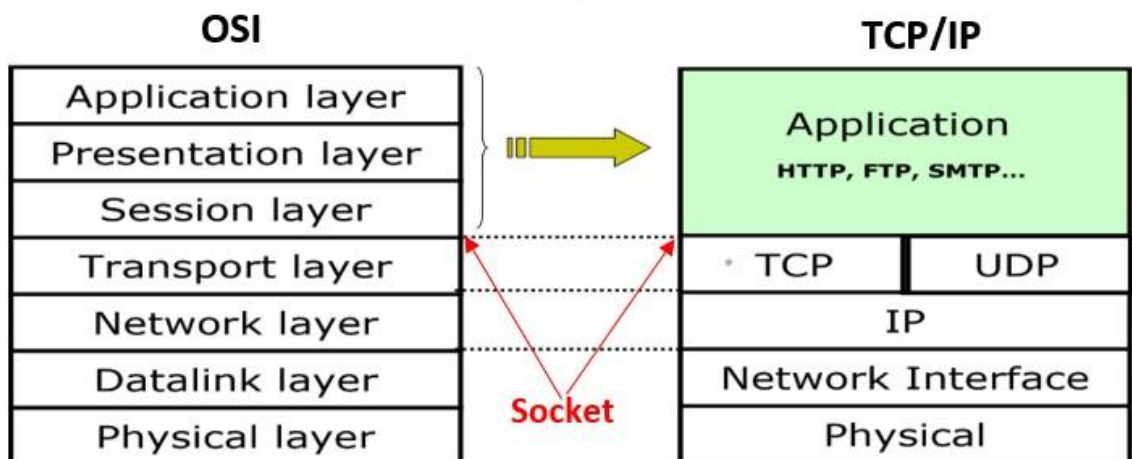
- Tắc nghẽn lưu lượng: Khi có quá nhiều client gửi request từ cùng 1 server sẽ dẫn đến kết nối chậm và gây tắc nghẽn.
- Chi phí: Bởi vì hệ thống mạng có sức mạnh lớn nên việc bảo trì và thiết lập thường khá cao mà không phải người dùng nào cũng chi trả được.
- Độ bền: Vì client – server là mạng tập trung nên server chỉ cần xảy ra sự cố hay bị nhiễu thì toàn bộ hệ thống mạng sẽ bị gián đoạn. Do vậy, các mạng client – server sẽ thiếu tính ổn định và độ bền.
- Phức tạp: Mô hình client-server có cấu trúc phức tạp và yêu cầu bảo trì thường xuyên. Vì vậy, luôn cần có quản trị viên mạng chuyên dụng để đảm bảo server hoạt động ổn định.

- Tài nguyên: Không phải tất cả tài nguyên trên server đều có thể truy cập từ client. Ví dụ, bạn không thể in tài liệu trực tiếp từ web hoặc chỉnh sửa bất kỳ thông tin nào trên ổ cứng của Client.

1.2.2 Lập trình socket với TCP

1.2.2.1 Socket

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



Hình 2 Mô hình OSI và TCP/IP

Dữ liệu bên gửi sẽ được đóng gói (Encapsulation) từ tầng trên đến tầng cuối là tầng vật lý (Physical Layer), sau đó nhờ tầng vật lý này chuyển dữ liệu đến tầng vật lý máy bên nhận, bên nhận tiến hành giải mã (decapsulation) gói dữ liệu 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 lý 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.

Tầng giao vận có 2 phương thức là TCP (Transmission 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 sử 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 trình 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.

Socket được hỗ trợ trên nhiều ngôn ngữ như C, Java, Pearl, Python,.... Sau đây là một ví dụ lập trình socket với Java.

1.2.2.2 Lập trình TCP Socket với Java

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.

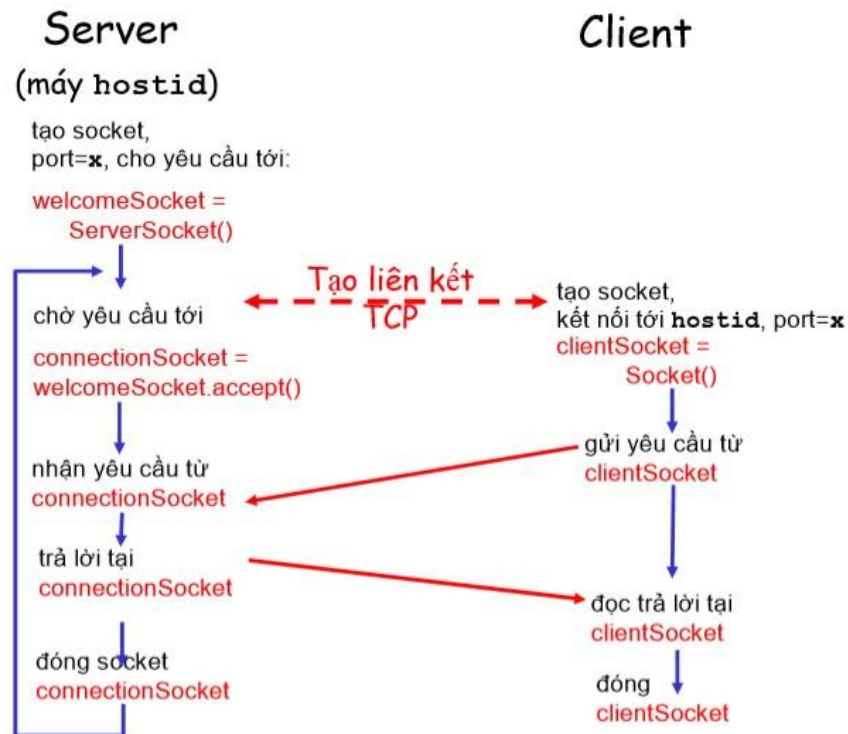
Tiếp đến phía client yêu cầu server bằng cách tạo một Socket TCP trên máy kèm với địa chỉ IP và port number của tiến trình tương ứng trên máy server. Khi client tạo Socket, client TCP tạo liên kết với server TCP và chờ chấp nhận kết nối từ server.

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 với Java

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 3 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 hoặc Java để chạy được ứng dụng client/server đã lập trình.

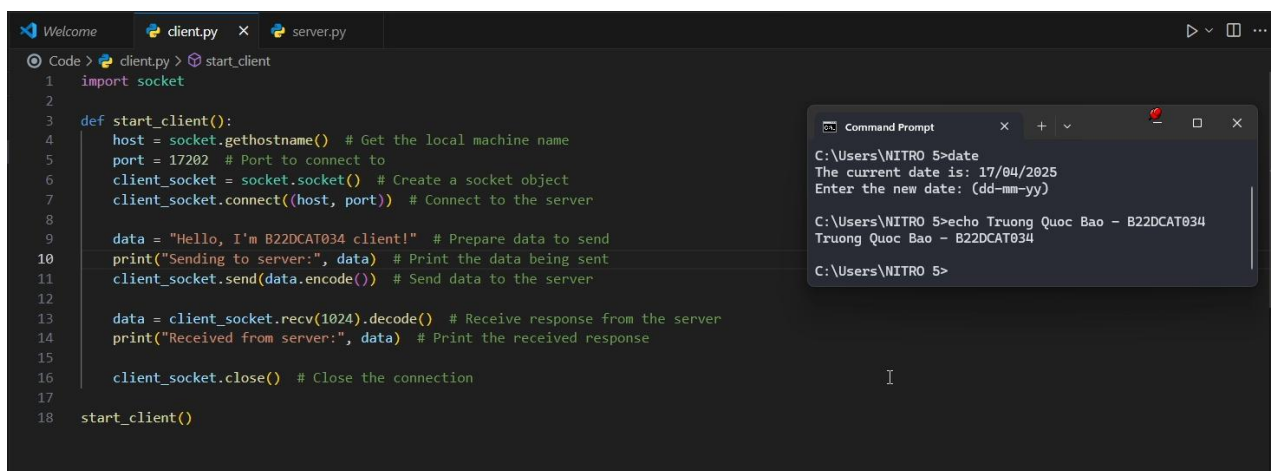
Phần mềm Wireshark

2.2 Các bước thực hiện

2.2.1 Lập trình client và server với TCP socket

Lập trình client

- Host = socket.gethostname() -> Lấy địa chỉ IP của máy đang chạy
- Port = 17202 -> Cổng được sử dụng để kết nối



The screenshot shows a VS Code editor with a file named `client.py` open. The code defines a `start_client()` function that connects to a server at `localhost` (port 17202) and sends a message. A terminal window is open, showing the execution of the client script, which prints the data being sent and received.

```
1 import socket
2
3 def start_client():
4     host = socket.gethostname() # Get the local machine name
5     port = 17202 # Port to connect to
6     client_socket = socket.socket() # Create a socket object
7     client_socket.connect((host, port)) # Connect to the server
8
9     data = "Hello, I'm B22DCAT034 client!" # Prepare data to send
10    print("Sending to server:", data) # Print the data being sent
11    client_socket.send(data.encode()) # Send data to the server
12
13    data = client_socket.recv(1024).decode() # Receive response from the server
14    print("Received from server:", data) # Print the received response
15
16    client_socket.close() # Close the connection
17
18 start_client()
```

Command Prompt output:

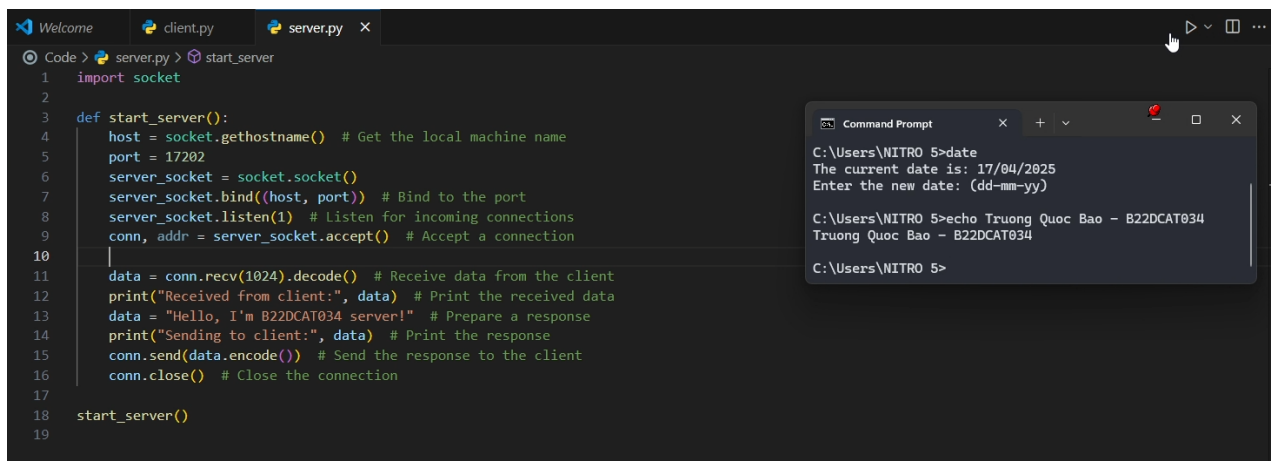
```
C:\Users\NITRO S>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\NITRO S>echo Truong Quoc Bao - B22DCAT034
Truong Quoc Bao - B22DCAT034

C:\Users\NITRO S>
```

Hình 4 Lập trình cho Client

Lập trình server



The screenshot shows a VS Code editor with a file named `server.py` open. The code defines a `start_server()` function that listens for incoming connections on port 17202 and responds to the client. A terminal window is open, showing the execution of the server script, which prints the data received from the client and the response sent back.

```
1 import socket
2
3 def start_server():
4     host = socket.gethostname() # Get the local machine name
5     port = 17202
6     server_socket = socket.socket()
7     server_socket.bind((host, port)) # Bind to the port
8     server_socket.listen(1) # Listen for incoming connections
9     conn, addr = server_socket.accept() # Accept a connection
10
11    data = conn.recv(1024).decode() # Receive data from the client
12    print("Received from client:", data) # Print the received data
13    data = "Hello, I'm B22DCAT034 server!" # Prepare a response
14    print("Sending to client:", data) # Print the response
15    conn.send(data.encode()) # Send the response to the client
16    conn.close() # Close the connection
17
18 start_server()
```

Command Prompt output:

```
C:\Users\NITRO S>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\NITRO S>echo Truong Quoc Bao - B22DCAT034
Truong Quoc Bao - B22DCAT034

C:\Users\NITRO S>
```

Hình 5 Lập trình cho Server

Chạy server sau đó chạy client

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

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”

```
C:\Users\NITRO 5\OneDrive\Desktop\Học\TTCS\TTCS_Res\Code>python server.py
Server started
Received from client: Hello, I'm B22DCAT034 client!
Sending to client: Hello, I'm B22DCAT034 server!
```

Hình 6 Khởi chạy Server

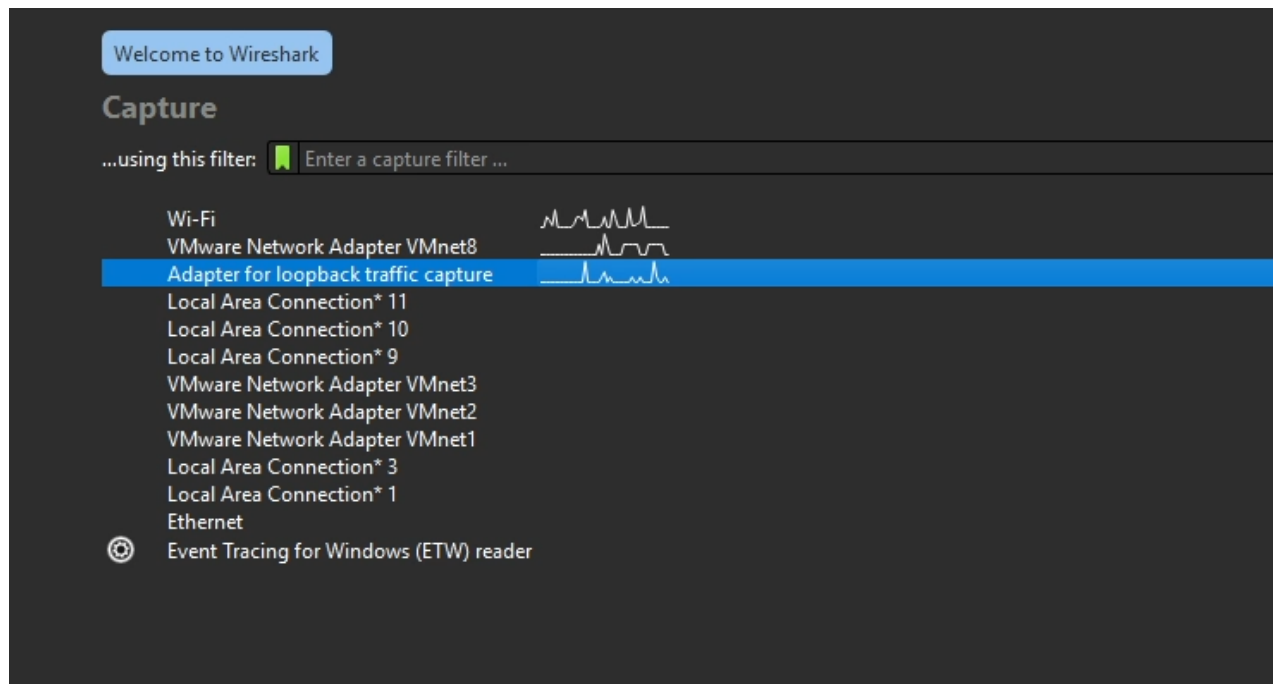
```
C:\Users\NITRO 5\OneDrive\Desktop\Học\TTCS\TTCS_Res\Code>python client.py
Client started
Sending to server: Hello, I'm B22DCAT034 client!
Received from server: Hello, I'm B22DCAT034 server!

C:\Users\NITRO 5\OneDrive\Desktop\Học\TTCS\TTCS_Res\Code>|
```

Hình 7 Khởi chạy Client

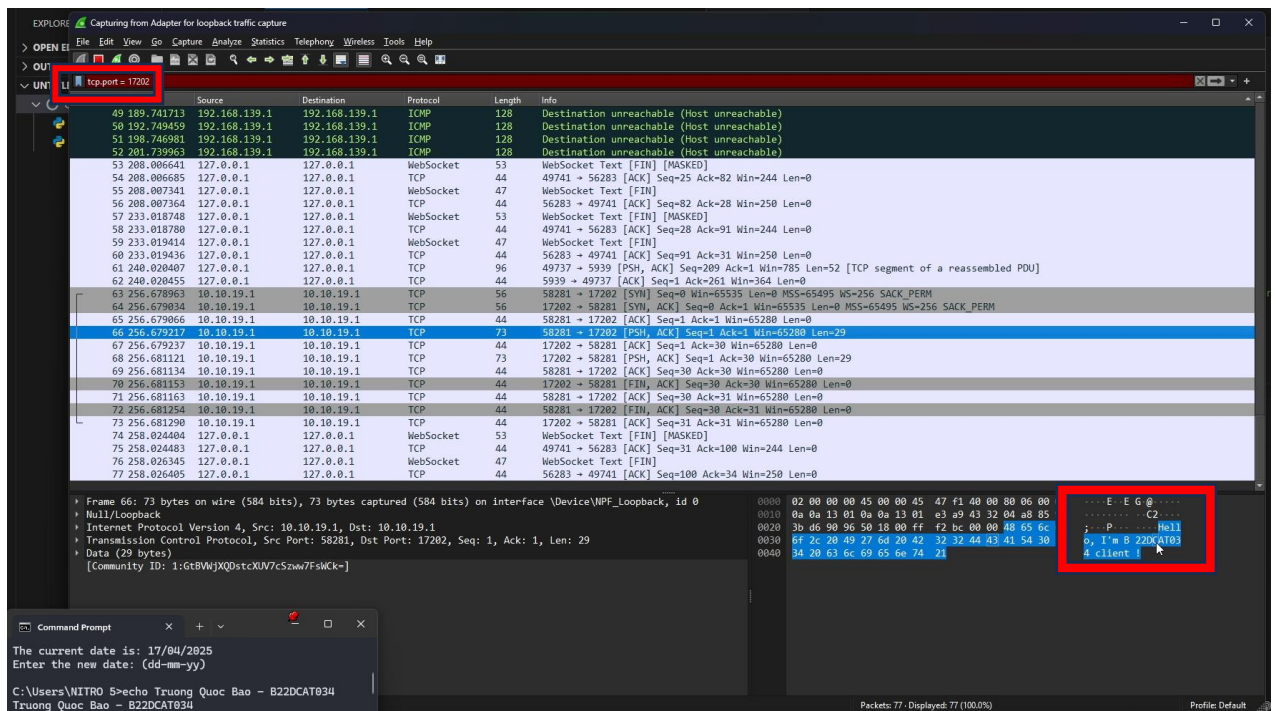
Sử dụng Wireshark để bắt các thông tin đã gửi từ client đến server và ngược lại

- Lựa chọn Adapter for loopback traffic capture
- Nhập bộ lọc tcp.port = 17202 để lọc các gói tin sử dụng giao thức TCP đi qua cổng 17202 đã cài đặt trước đó.

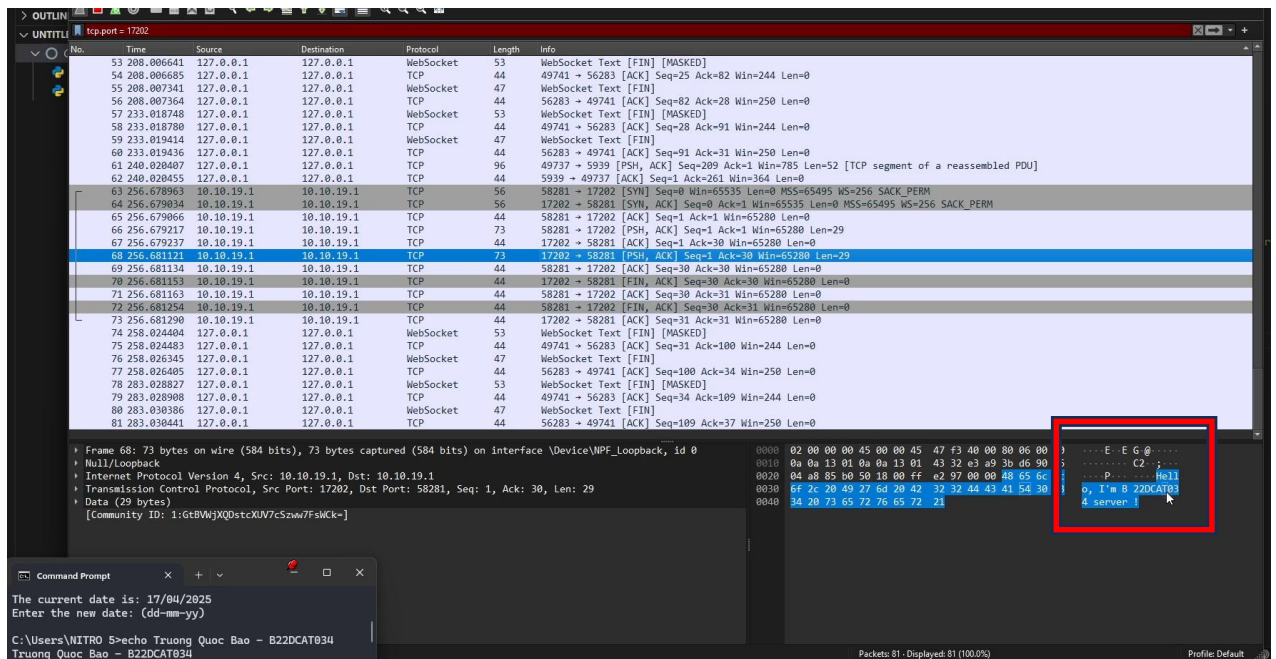


Hình 8 Khởi động Wireshark để bắt các gói tin

- Wireshark sẽ bắt và đọc đc các gói tin chứa thông điệp được gửi giữa Client và Server.



Hình 9 Wireshark bắt được thông điệp 1

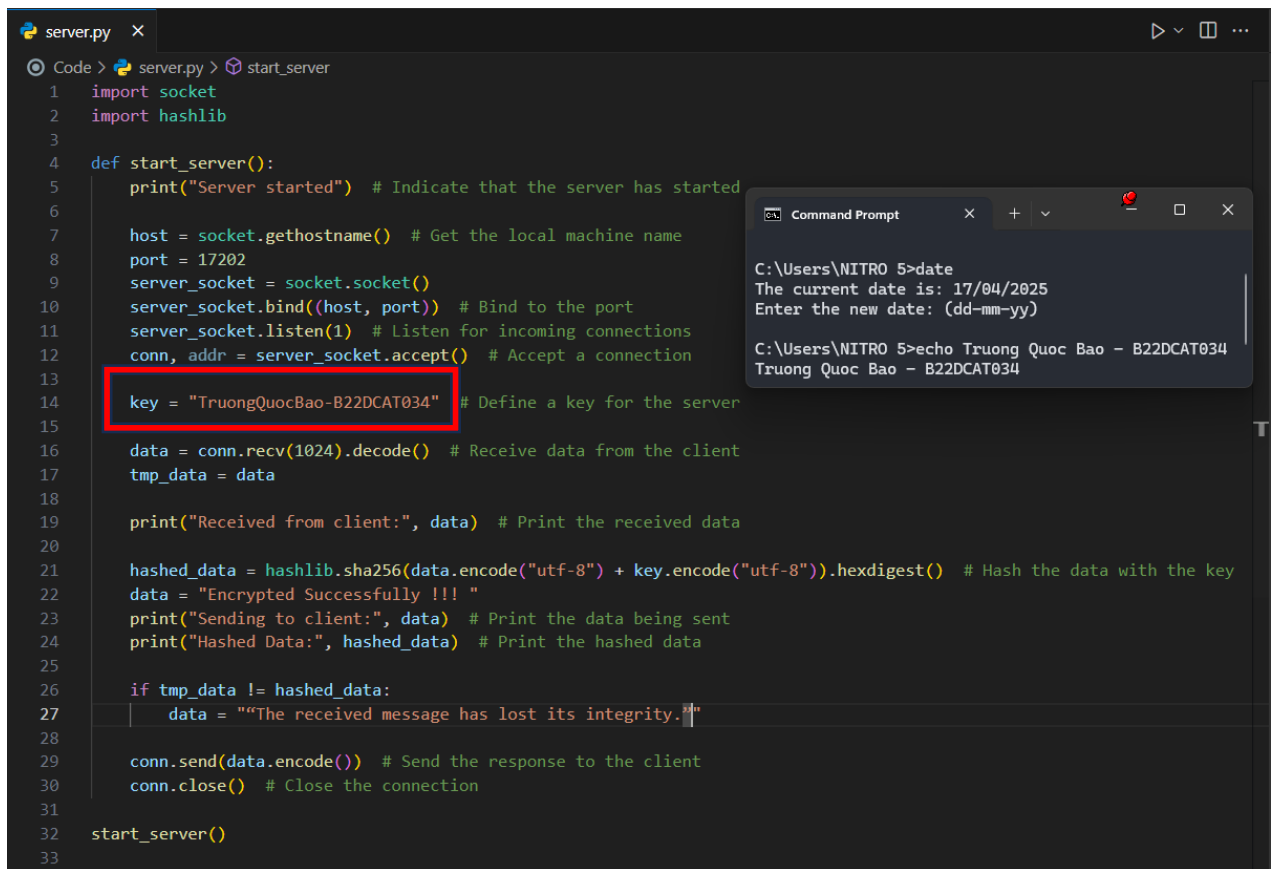


Hình 10 Wireshark bắt được thông điệp 2

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.

Trường hợp cùng giá trị key.



The screenshot shows a Python IDE with a file named `server.py`. The code defines a `start_server()` function that listens for a connection, receives data, and hashes it using SHA256 with a key. The key is defined as `key = "TruongQuocBao-B22DCAT034"` and is highlighted with a red box. A terminal window is open, showing the execution of the script. The terminal output shows the server starting, receiving data, and sending the hashed data back to the client.

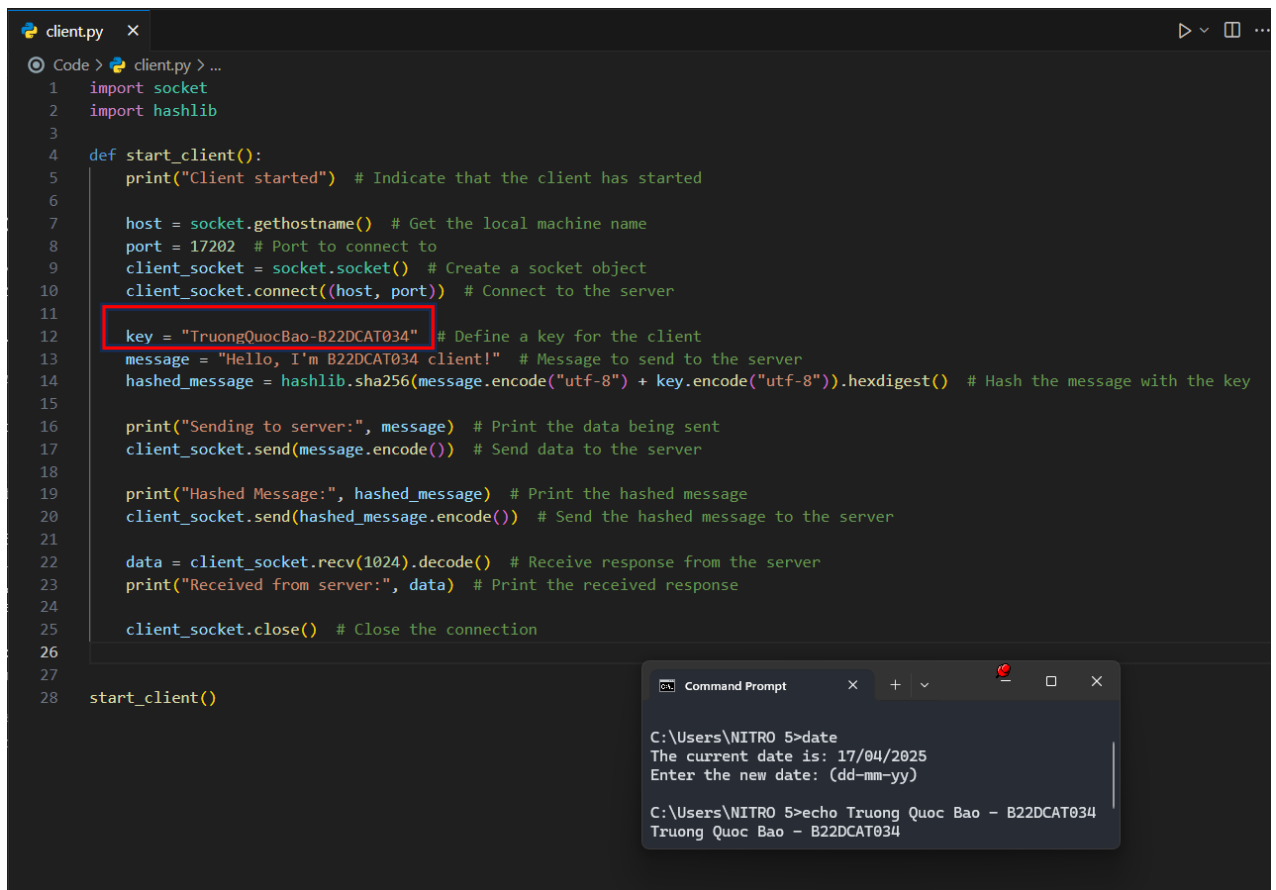
```
server.py
Code > server.py > start_server
1 import socket
2 import hashlib
3
4 def start_server():
5     print("Server started") # Indicate that the server has started
6
7     host = socket.gethostname() # Get the local machine name
8     port = 17202
9     server_socket = socket.socket()
10    server_socket.bind((host, port)) # Bind to the port
11    server_socket.listen(1) # Listen for incoming connections
12    conn, addr = server_socket.accept() # Accept a connection
13
14    key = "TruongQuocBao-B22DCAT034" # Define a key for the server
15
16    data = conn.recv(1024).decode() # Receive data from the client
17    tmp_data = data
18
19    print("Received from client:", data) # Print the received data
20
21    hashed_data = hashlib.sha256(data.encode("utf-8") + key.encode("utf-8")).hexdigest() # Hash the data with the key
22    data = "Encrypted Successfully !!! "
23    print("Sending to client:", data) # Print the data being sent
24    print("Hashed Data:", hashed_data) # Print the hashed data
25
26    if tmp_data != hashed_data:
27        data = "The received message has lost its integrity."
28
29    conn.send(data.encode()) # Send the response to the client
30    conn.close() # Close the connection
31
32 start_server()
33
```

Command Prompt

```
C:\Users\NITRO 5>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\NITRO 5>echo Truong Quoc Bao - B22DCAT034
Truong Quoc Bao - B22DCAT034
```

Hình 11 Lập trình Server với key



The image shows a code editor window titled 'client.py' with the following Python code:

```
1 import socket
2 import hashlib
3
4 def start_client():
5     print("Client started") # Indicate that the client has started
6
7     host = socket.gethostname() # Get the local machine name
8     port = 17202 # Port to connect to
9     client_socket = socket.socket() # Create a socket object
10    client_socket.connect((host, port)) # Connect to the server
11
12    key = "TruongQuocBao-B22DCAT034" # Define a key for the client
13    message = "Hello, I'm B22DCAT034 client!" # Message to send to the server
14    hashed_message = hashlib.sha256(message.encode("utf-8") + key.encode("utf-8")).hexdigest() # Hash the message with the key
15
16    print("Sending to server:", message) # Print the data being sent
17    client_socket.send(message.encode()) # Send data to the server
18
19    print("Hashed Message:", hashed_message) # Print the hashed message
20    client_socket.send(hashed_message.encode()) # Send the hashed message to the server
21
22    data = client_socket.recv(1024).decode() # Receive response from the server
23    print("Received from server:", data) # Print the received response
24
25    client_socket.close() # Close the connection
26
27
28 start_client()
```

Below the code editor, a Command Prompt window shows the execution of the client script:

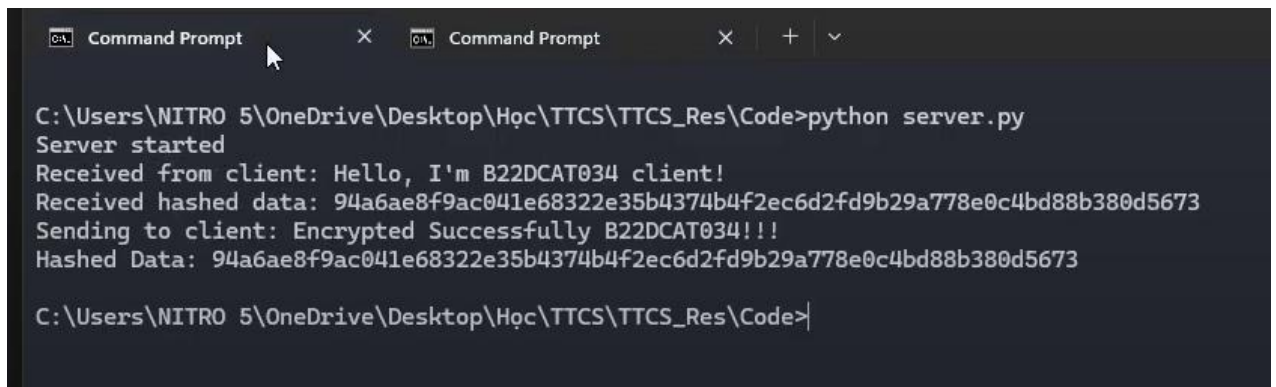
```
C:\Users\NITRO 5>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)

C:\Users\NITRO 5>echo Truong Quoc Bao - B22DCAT034
Truong Quoc Bao - B22DCAT034
```

Hình 12 Lập trình Client với key

Khởi chạy Server và Client

- Cả 2 bên Server và Client đều nhận được cùng một mã hash do có 2 key giống nhau



The image shows a Command Prompt window with the following output:

```
C:\Users\NITRO 5\OneDrive\Desktop\Hoc\TTCS\TTCS_Res\Code>python server.py
Server started
Received from client: Hello, I'm B22DCAT034 client!
Received hashed data: 94a6ae8f9ac041e68322e35b4374b4f2ec6d2fd9b29a778e0c4bd88b380d5673
Sending to client: Encrypted Successfully B22DCAT034!!!
Hashed Data: 94a6ae8f9ac041e68322e35b4374b4f2ec6d2fd9b29a778e0c4bd88b380d5673

C:\Users\NITRO 5\OneDrive\Desktop\Hoc\TTCS\TTCS_Res\Code>
```

Hình 13 Khởi chạy Server


```

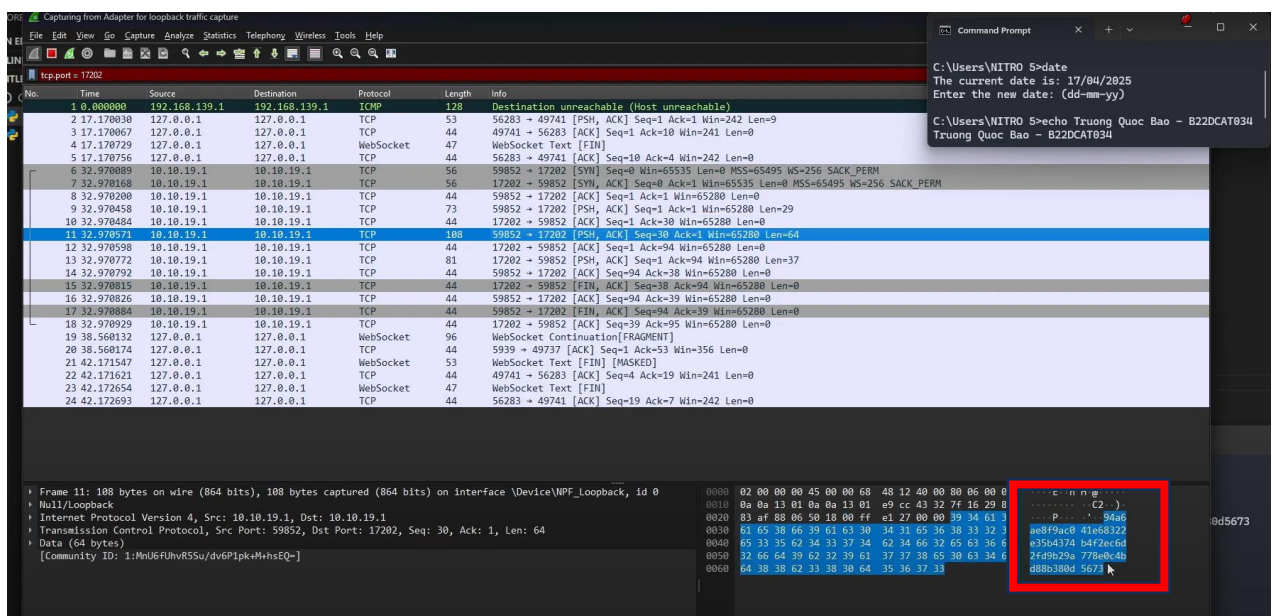
C:\Users\NITRO 5\OneDrive\Desktop\Hoc\TTCS\TTCS_Res\Code>python client.py
Client started
Sending to server: Hello, I'm B22DCAT034 client!
Hashed Message: 94a6ae8f9ac041e68322e35b4374b4f2ec6d2fd9b29a778e0c4bd88b380d5673
Received from server: Encrypted Successfully B22DCAT034!!!

C:\Users\NITRO 5\OneDrive\Desktop\Hoc\TTCS\TTCS_Res\Code>

```

Hình 14 Khởi chạy Client

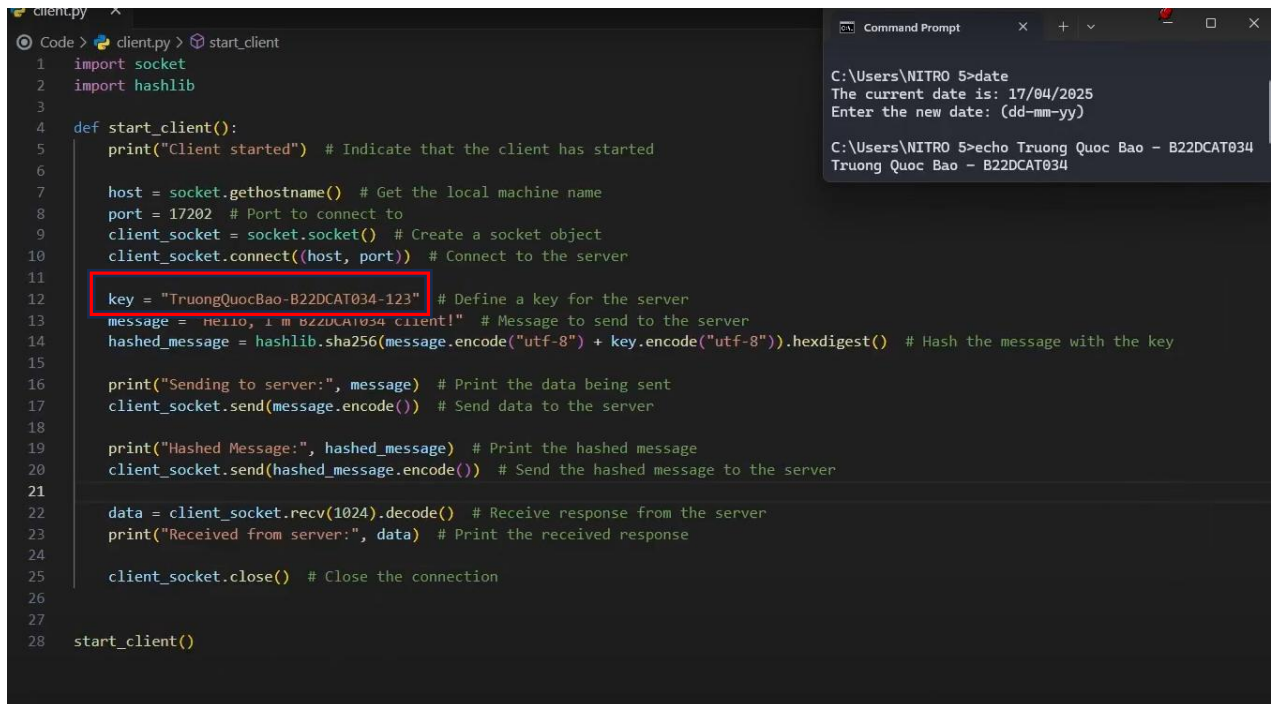
- Wireshark bắt được các gói tin và thông điệp mã hóa được gửi



Hình 15 Wireshark bắt được thông điệp đã mã hóa

Trường hợp khác giá trị key

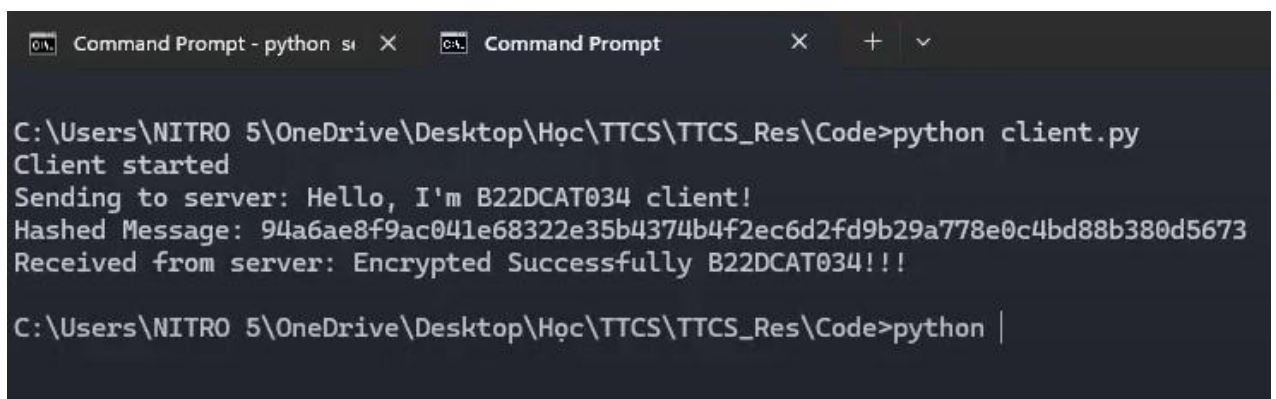
Thay đổi giá trị key tại client và thực hiện gửi lại, nếu không đáp ứng tính toàn vẹn cần thông báo: “The received message has lost its integrity.”



```
client.py
Code > client.py start_client
1 import socket
2 import hashlib
3
4 def start_client():
5     print("Client started") # Indicate that the client has started
6
7     host = socket.gethostname() # Get the local machine name
8     port = 17202 # Port to connect to
9     client_socket = socket.socket() # Create a socket object
10    client_socket.connect((host, port)) # Connect to the server
11
12    key = "TruongQuocBao-B22DCAT034-123" # Define a key for the server
13    message = "hello, i m B22DCAT034 client!" # Message to send to the server
14    hashed_message = hashlib.sha256(message.encode("utf-8") + key.encode("utf-8")).hexdigest() # Hash the message with the key
15
16    print("Sending to server:", message) # Print the data being sent
17    client_socket.send(message.encode()) # Send data to the server
18
19    print("Hashed Message:", hashed_message) # Print the hashed message
20    client_socket.send(hashed_message.encode()) # Send the hashed message to the server
21
22    data = client_socket.recv(1024).decode() # Receive response from the server
23    print("Received from server:", data) # Print the received response
24
25    client_socket.close() # Close the connection
26
27
28 start_client()
```

```
Command Prompt
C:\Users\NITRO 5>date
The current date is: 17/04/2025
Enter the new date: (dd-mm-yy)
C:\Users\NITRO 5>echo Truong Quoc Bao - B22DCAT034
Truong Quoc Bao - B22DCAT034
```

Hình 16 Thay đổi giá trị key của Client



```
Command Prompt - python s4 X Command Prompt X + v
C:\Users\NITRO 5\OneDrive\Desktop\Hoc\TTCS\TTCS_Res\Code>python client.py
Client started
Sending to server: Hello, I'm B22DCAT034 client!
Hashed Message: 94a6ae8f9ac041e68322e35b4374b4f2ec6d2fd9b29a778e0c4bd88b380d5673
Received from server: Encrypted Successfully B22DCAT034!!!
C:\Users\NITRO 5\OneDrive\Desktop\Hoc\TTCS\TTCS_Res\Code>python |
```

Hình 17 Client truyền thông điệp

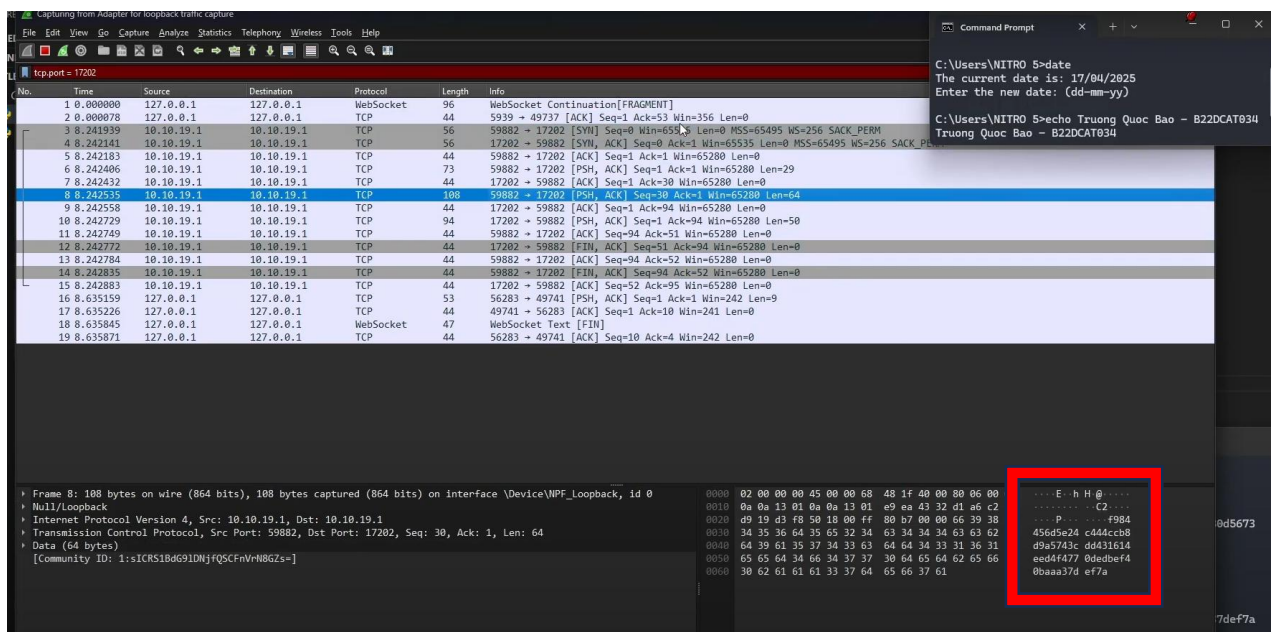
Server nhận được phản hồi “Thông điệp đã mã hóa thành công”


```
C:\Users\NITRO 5\OneDrive\Desktop\Học\TTCS\TTCS_Res\Code>python client.py
Client started
Sending to server: Hello, I'm B22DCAT034 client!
Hashed Message: 94a6ae8f9ac041e68322e35b4374b4f2ec6d2fd9b29a778e0c4bd88b380d5673
Received from server: Encrypted Successfully B22DCAT034!!!

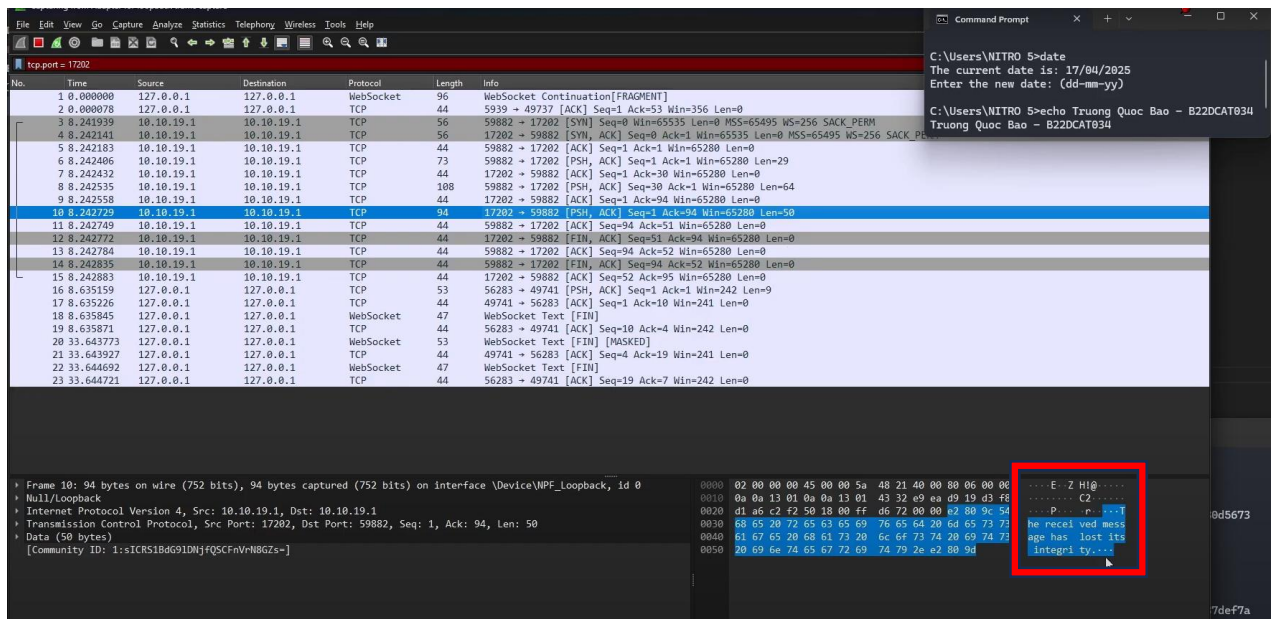
C:\Users\NITRO 5\OneDrive\Desktop\Học\TTCS\TTCS_Res\Code>python client.py
Client started
Sending to server: Hello, I'm B22DCAT034 client!
Hashed Message: f984456d5e24c444ccb8d9a5743cdd431614eed4f4770dedbef40baaa37def7a
Received from server: "The received message has lost its integrity."
```

Hình 18 Thông điệp đến server đã bị khác đi

- Server và Client trả về 2 mã hash khác nhau do 2 key khác nhau
- Bắt được các bản tin trao đổi giữa client và server trong Wireshark



Hình 19 Wireshark bắt được mã hash của thông điệp



Hình 20 Thông báo thông điệp đã mất mát

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