

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



Môn học: Thực Tập Cơ Sở

Báo Cáo Bài Thực Hành 15

Lập trình Client/Server để trao đổi thông tin

Họ và tên: Trần Thị Thu Phương

Mã sinh viên: B21DCAT151

Nhóm môn học: 04

Giảng viên: Đinh Trường Duy

Hà Nội, 4/2024

Mục lục

1. Mục đích	3
2. Nội dung thực hành	3
2.1. Cơ sở lý thuyết: Tìm hiểu về các khái niệm liên quan đến lập trình socket với TCP	3
2.2. Các bước thực hiện	7
2.2.1. Chuẩn bị môi trường	7
2.2.2. Lập trình client và server với TCP socket	7
2.2.3. Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi.....	10
3. Kết luận	18
4. Tài liệu tham khảo	18

Danh mục hình ảnh

Socket hoạt động thông qua các tầng TCP hoặc TCP Layer định danh ứng dụng, từ đó truyền dữ liệu thông qua sự ràng buộc với một cổng port	3
Chức năng của socket là kết nối giữa client và server thông qua TCP/IP và UDP để truyền và nhận giữ liệu qua Internet	4
Lập trình và chạy Server.....	8
Lập trình và chạy Client	8
Bắt thông tin được gửi từ Server đến Client.....	9
Bắt thông tin được gửi từ Client đến Server.....	9
Code của Server.....	10
Kết quả khi chạy	11
Code Client	11
Kết quả khi chạy Client	12
Client gửi thông điệp bản rõ đến Server.....	12
Client gửi thông điệp + key đã được mã hóa SHA512	13
Server gửi phản hồi đến Client sau khi nhận thông điệp + key không thay đổi	13
Thay đổi key ở Client → Thông điệp mã hash thay đổi.....	14
Bên Server giữ nguyên	15
Bên Server: mã hash của thông điệp đã bị thay đổi → Không toàn vẹn dữ liệu.....	16
Kết quả, bên Client nhận được thông báo rằng dữ liệu đã bị mất mát	16
Khi truyền từ Client → Server, Mã hash đã bị thay đổi.....	17
Server phản hồi Client: dữ liệu đã mất mát	17

1. Mục đích

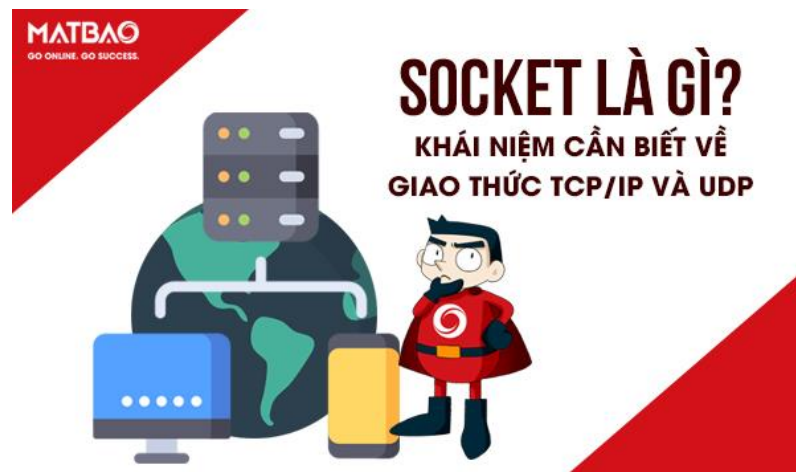
Sinh viên hiểu về cơ chế client/server và có thể tự lập trình client/server dựa trên socket, sau đó thực hiện ca đặt giao thức đơn giản để trao đổi thông tin an toàn.

2. Nội dung thực hành

2.1. Cơ sở lý thuyết: Tìm hiểu về các khái niệm liên quan đến lập trình socket với TCP

a. 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.



Socket hoạt động thông qua các tầng TCP hoặc TCP Layer định danh ứng dụng, từ đó truyền dữ liệu thông qua sự ràng buộc với một cổng port

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.

b. 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.

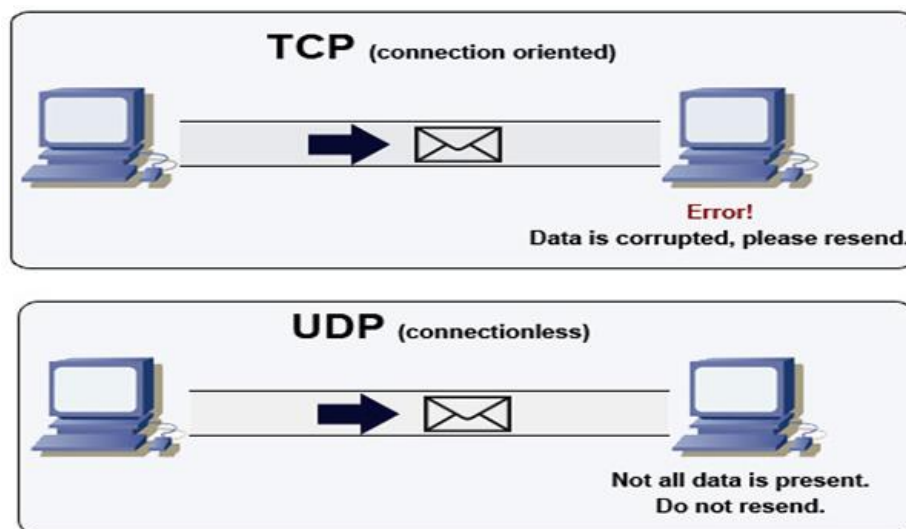
Phân phối dữ liệu: Sử dụng socket, dữ liệu có thể được phân phối từ một nguồn tới nhiều đích hoặc ngược lại. Điều này cho phép triển khai các ứng dụng mạng phức tạp như streaming video, trò chơi trực tuyến, và các ứng dụng đa người dùng.

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.

c. Điều kiện để Socket hoạt động?



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

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.

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

- 2 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.

d. 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.

e. 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).

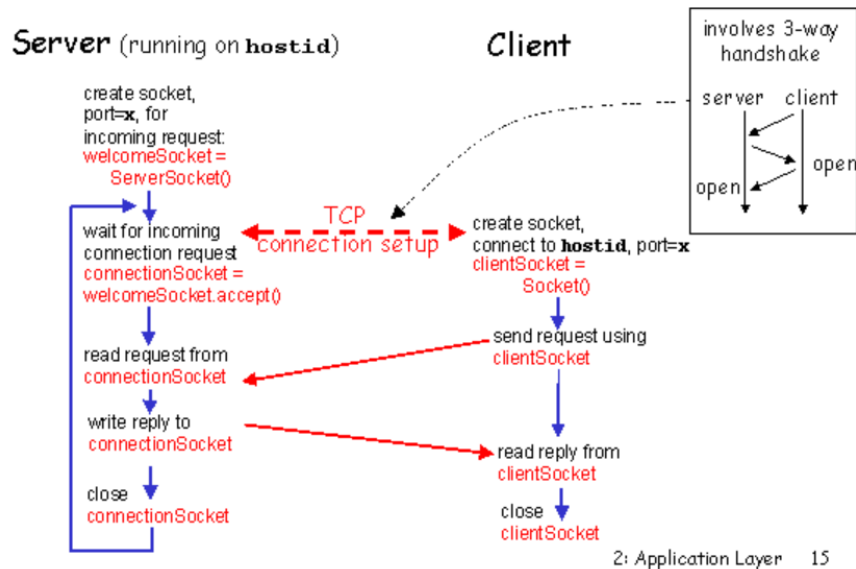
Để có thể tương tác với nhau và máy chủ có thể nhận liên lạc từ máy khách thì máy chủ phải luôn sẵn sàng. Điều này có 2 nghĩa, thứ nhất, giống như trường hợp của UDP, một tiến trình của máy chủ phải được chạy trước khi máy khách khởi tạo liên lạc đến. Thứ hai, chương trình chủ phải tạo ra 1 socket để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

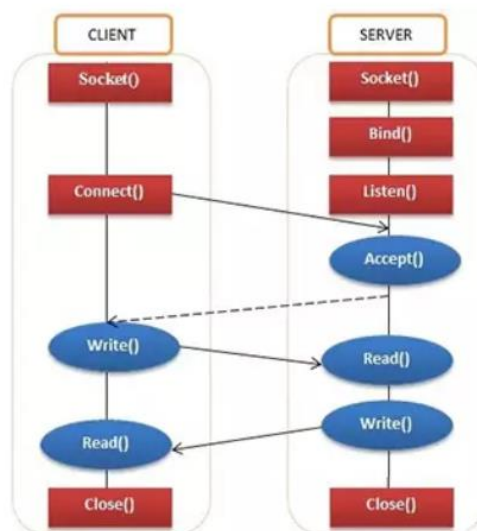
Khi socket của tiến trình khách vừa được tạo, TCP trên máy khách sẽ tiến hành thực hiện quá trình bắt tay 3 bước và thiết lập kết nối TCP tới máy chủ.

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



Mô tả quá trình:



1. Trước tiên chúng ta sẽ tạo ra một máy chủ bằng cách mở một socket - `Socket()`

2. Sau đó chúng ta sẽ liên kết nó với một host hoặc một máy và một port - Bind()
3. Tiếp theo server sẽ bắt đầu lắng nghe trên port đó - Listen()
4. Yêu cầu kết nối từ client được gửi tới server - Connect()
5. Server sẽ accept yêu cầu từ client và sau đó kết nối được thiết lập - Accept()
6. Bây giờ cả hai đều có thể gửi và nhận tin tại thời điểm đó - Read() / Write()
7. Và cuối cùng khi hoàn thành chúng có thể đóng kết nối - Close()

2.2. Các bước thực hiện

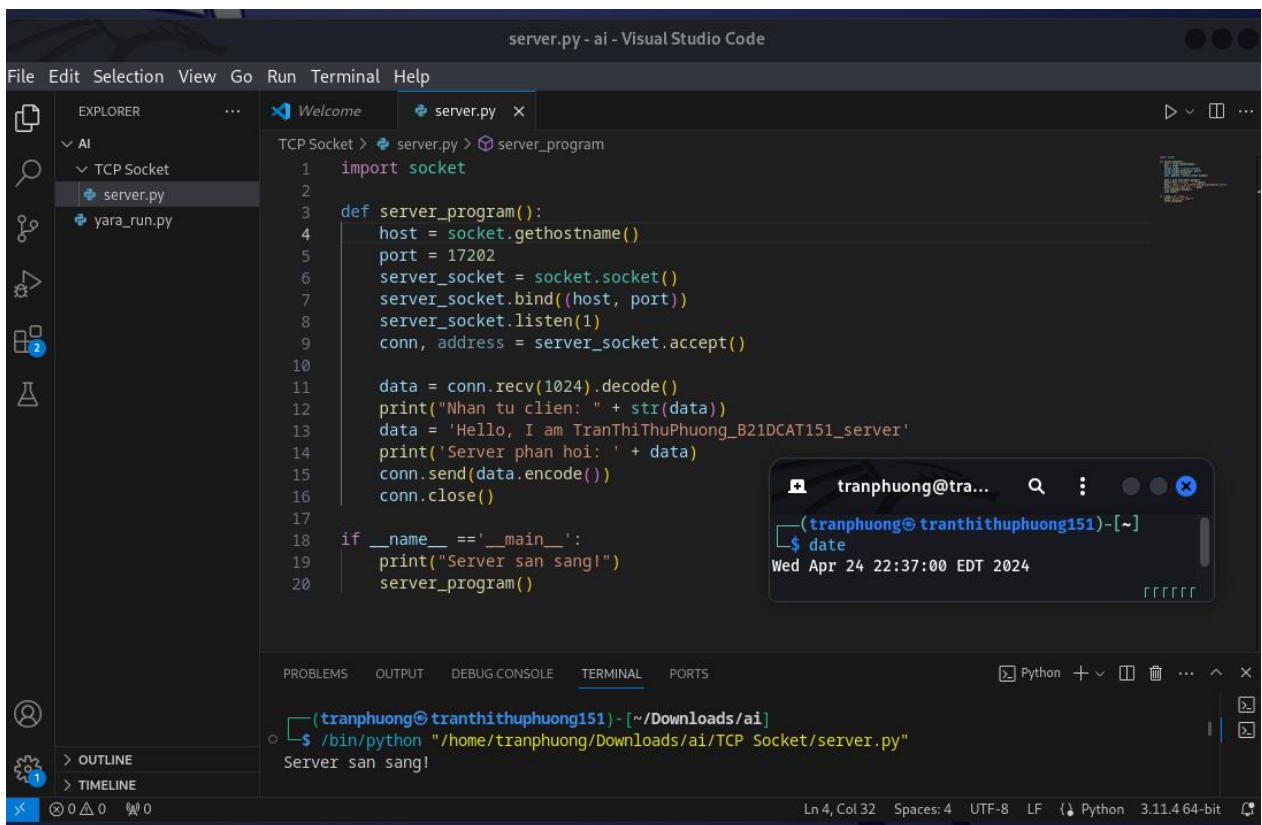
2.2.1. Chuẩn bị môi trường

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

2.2.2. 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 TranThiThuPhuong_B21DCAT151_client.”
- Server nhận được hiển thị thông điệp nhận được và gửi lại client thông điệp: server gửi lại “Hello, I am TranThiThuPhuong_B21DCAT151_server”

Bài 15: Lập trình Client/Server để trao đổi thông tin an toàn

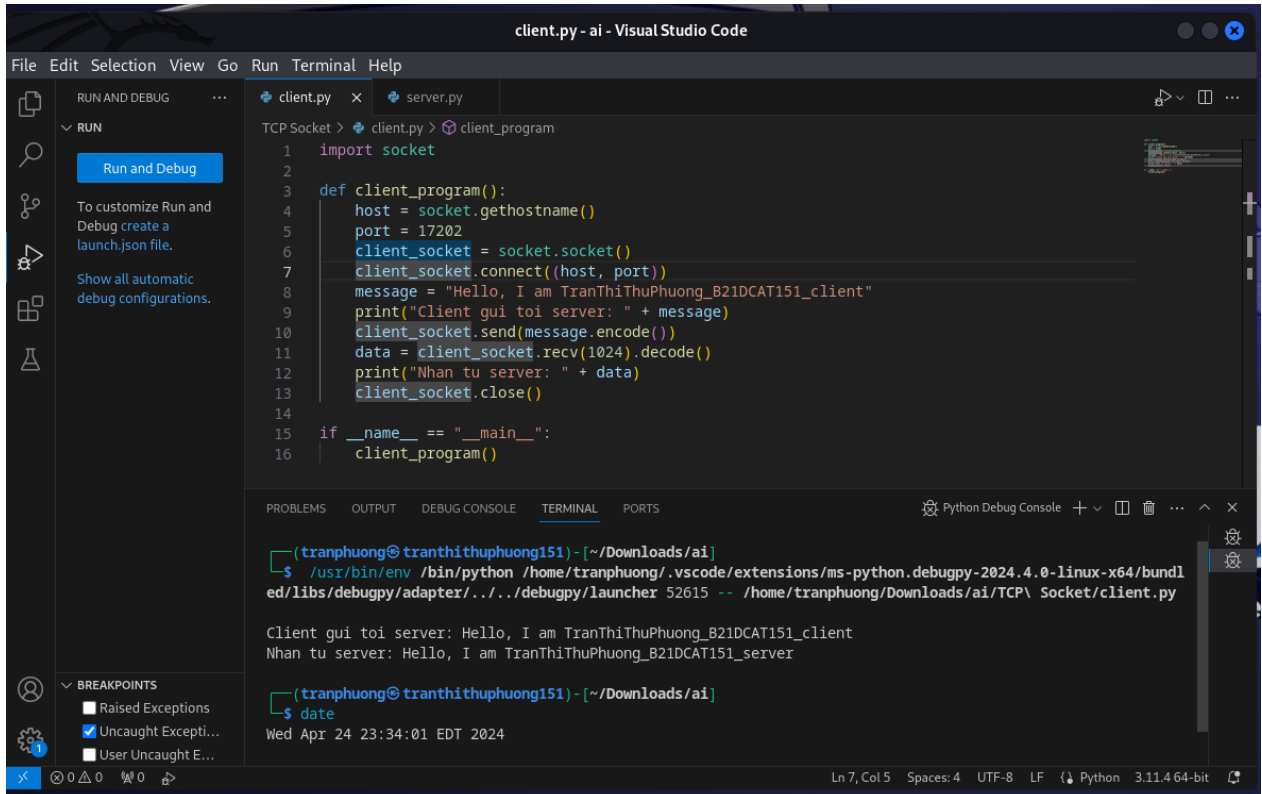


The screenshot shows the Visual Studio Code editor with a file named `server.py` open. The code is a Python script that implements a simple TCP server. It imports the `socket` module, defines a `server_program()` function, and uses `socket.gethostname()` and `socket.socket()` to create a server socket. The server binds to the local host and port 17202, listens for incoming connections, and when a connection is accepted, it receives data from the client, prints it, and sends a response back to the client. The response is "Hello, I am TranThiThuPhuong_B21DCAT151_server". The server then closes the connection. The `if __name__ == '__main__':` block calls `server_program()`.

```
1 import socket
2
3 def server_program():
4     host = socket.gethostname()
5     port = 17202
6     server_socket = socket.socket()
7     server_socket.bind((host, port))
8     server_socket.listen(1)
9     conn, address = server_socket.accept()
10
11     data = conn.recv(1024).decode()
12     print("Nhan tu clien: " + str(data))
13     data = 'Hello, I am TranThiThuPhuong_B21DCAT151_server'
14     print('Server phan hoi: ' + data)
15     conn.send(data.encode())
16     conn.close()
17
18 if __name__ == '__main__':
19     print("Server san sang!")
20     server_program()
```

The terminal window shows the command `/bin/python "/home/tranphuong/Downloads/ai/TCP Socket/server.py"` being executed, and the output `Server san sang!`.

Lập trình và chạy Server



The screenshot shows the Visual Studio Code editor with a file named `client.py` open. The code is a Python script that implements a simple TCP client. It imports the `socket` module, defines a `client_program()` function, and uses `socket.gethostname()` and `socket.socket()` to create a client socket. The client connects to the local host and port 17202, sends a message to the server, receives data from the server, prints it, and then closes the connection. The message is "Hello, I am TranThiThuPhuong_B21DCAT151_client". The `if __name__ == '__main__':` block calls `client_program()`.

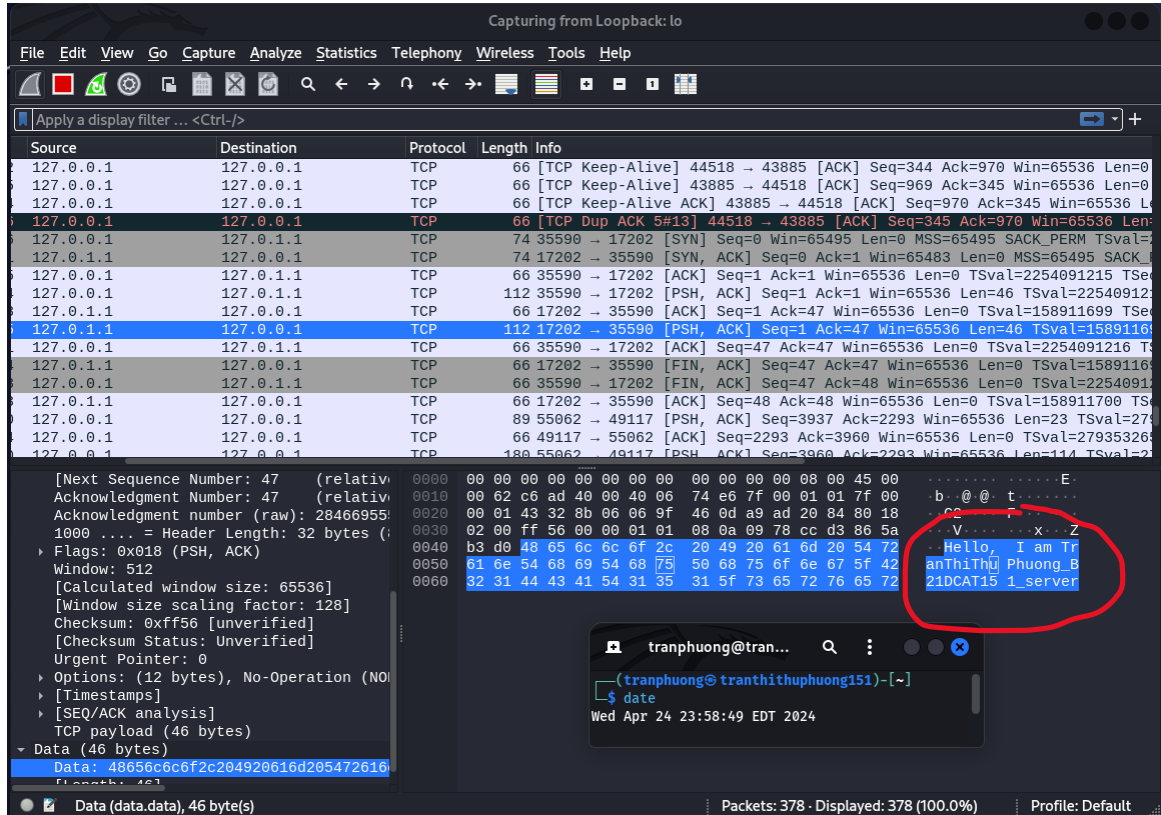
```
1 import socket
2
3 def client_program():
4     host = socket.gethostname()
5     port = 17202
6     client_socket = socket.socket()
7     client_socket.connect((host, port))
8     message = "Hello, I am TranThiThuPhuong_B21DCAT151_client"
9     print("Client gui toi server: " + message)
10    client_socket.send(message.encode())
11    data = client_socket.recv(1024).decode()
12    print("Nhan tu server: " + data)
13    client_socket.close()
14
15 if __name__ == '__main__':
16    client_program()
```

The terminal window shows the command `/usr/bin/env /bin/python /home/tranphuong/.vscode/extensions/ms-python.debugpy-2024.4.0-linux-x64/bundled/libs/debugpy/adapters/.../debugpy/launcher 52615 -- /home/tranphuong/Downloads/ai/TCP Socket/client.py` being executed. The output shows the client sending a message to the server and receiving a response: `Client gui toi server: Hello, I am TranThiThuPhuong_B21DCAT151_client` and `Nhan tu server: Hello, I am TranThiThuPhuong_B21DCAT151_server`.

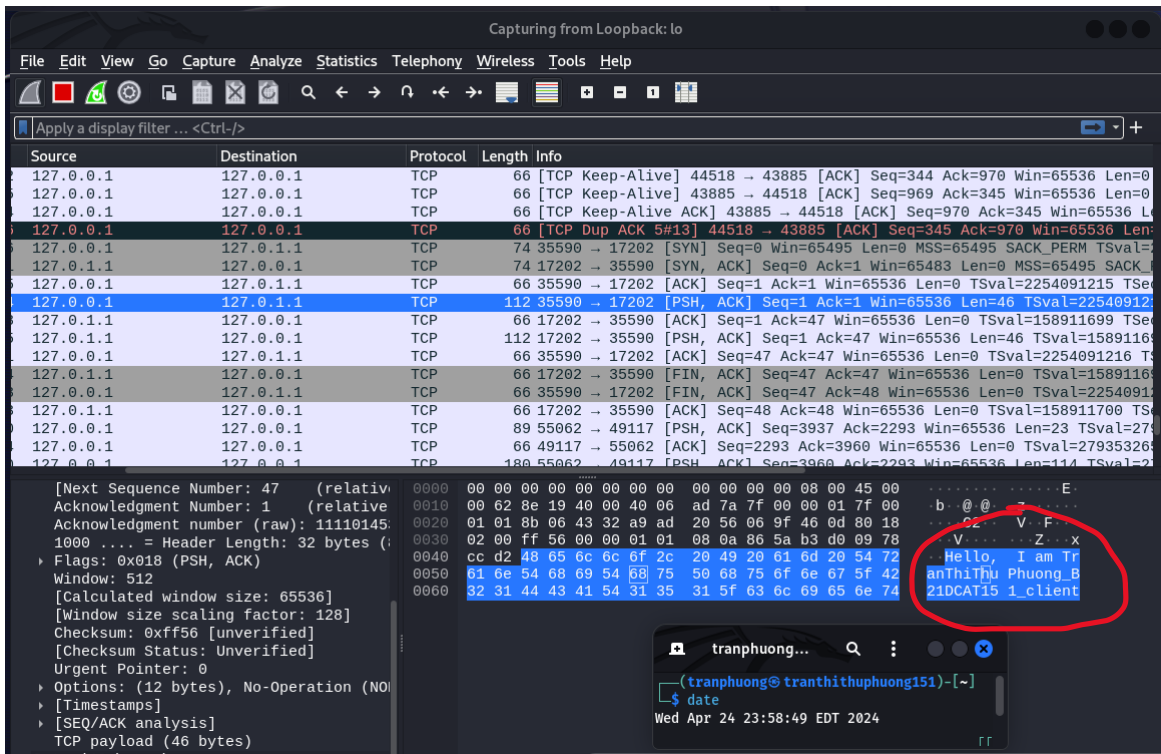
Lập trình và chạy Client

Bài 15: Lập trình Client/Server để trao đổi thông tin an toàn

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



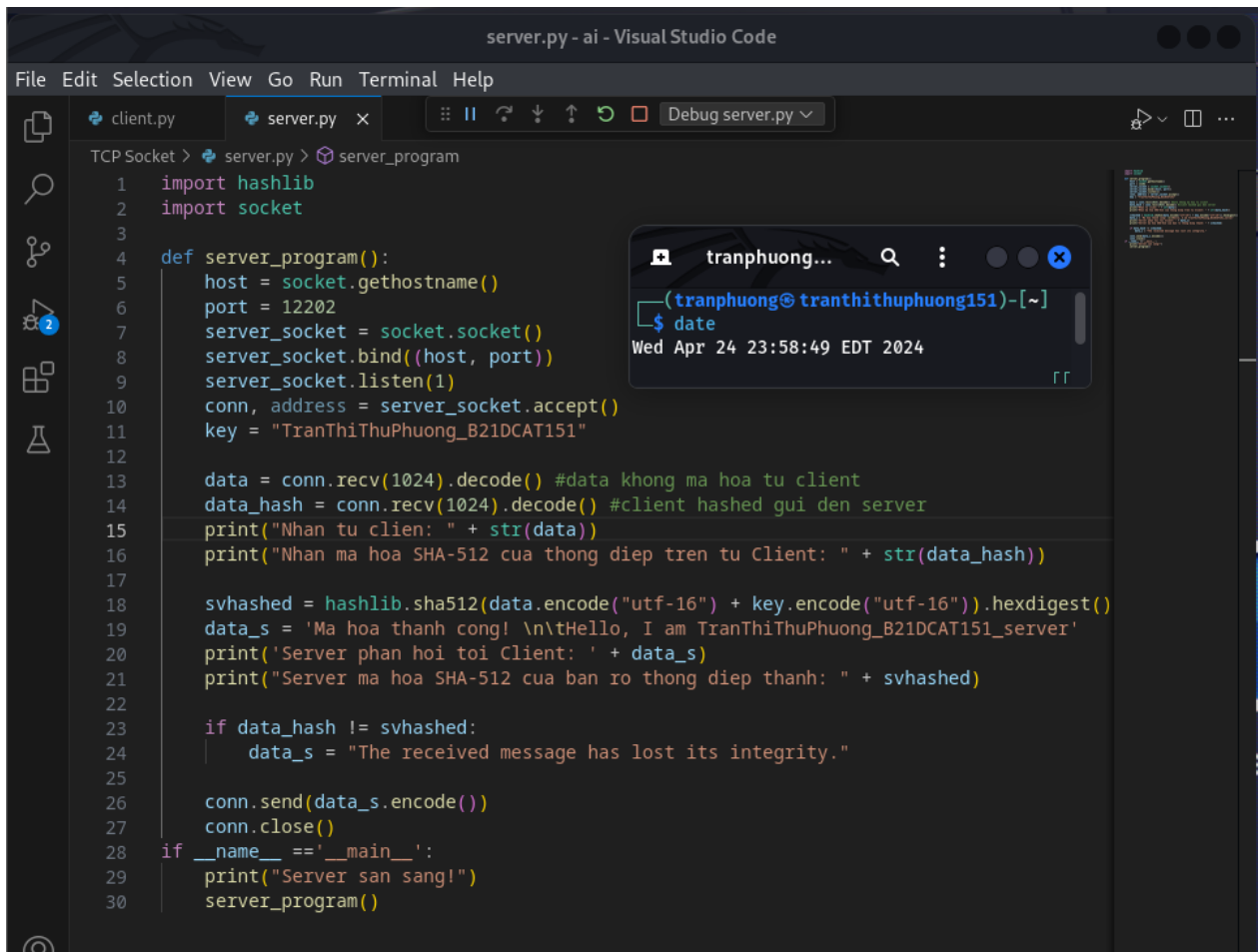
Bắt thông tin được gửi từ Server đến Client



Bắt thông tin được gửi từ Client đến Server

2.2.3. 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.



The screenshot shows the Visual Studio Code editor with a file named `server.py` open. The code is a Python script for a server program that listens on port 12202. It receives a message from a client, calculates its SHA-512 hash combined with a key, and sends the original message and the hash back to the client. A terminal window is open, showing the command prompt of a user named `tranphuong...` at `tranthithuphuong151`. The terminal shows the command `date` being executed, resulting in the output `Wed Apr 24 23:58:49 EDT 2024`.

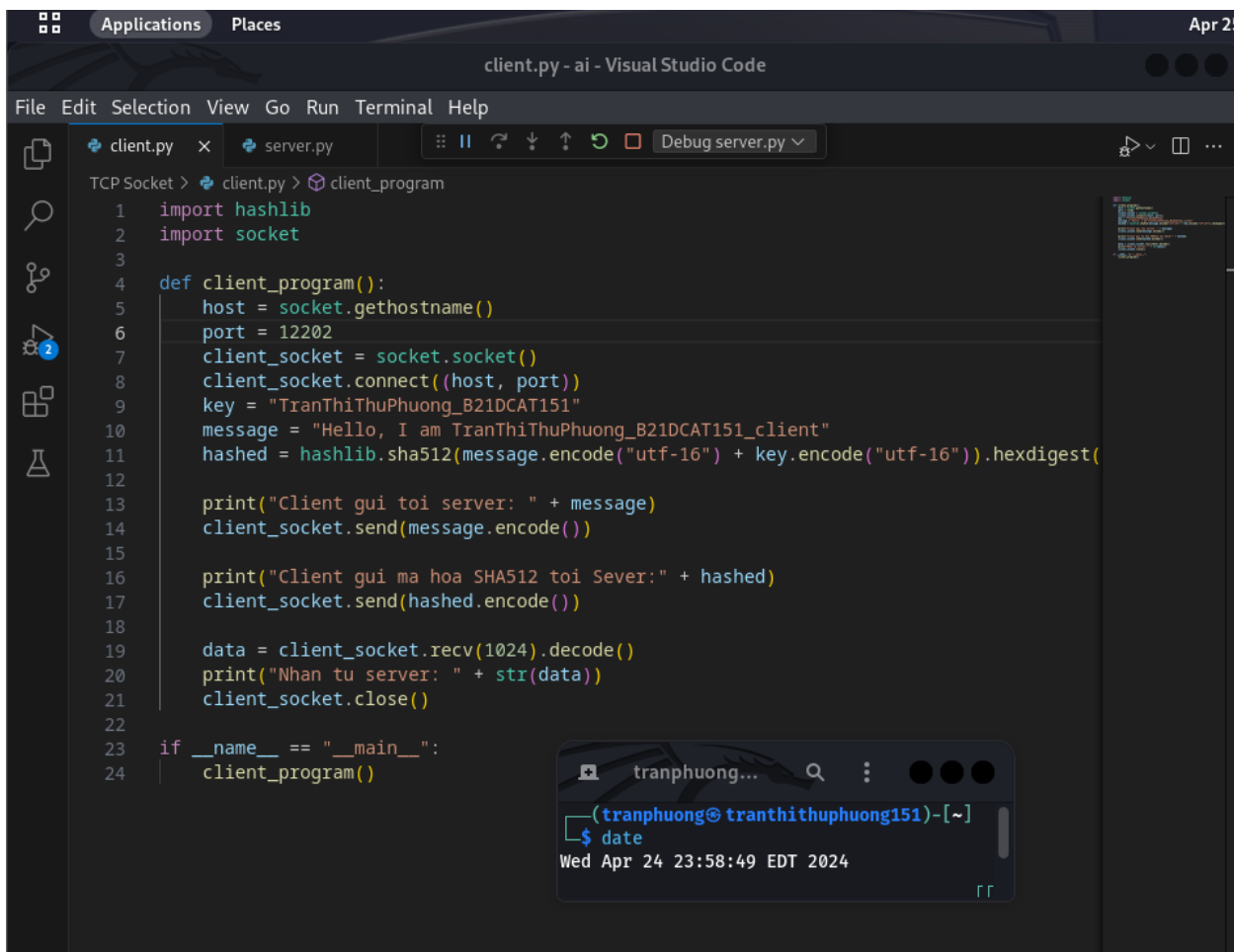
```
server.py - ai - Visual Studio Code
File Edit Selection View Go Run Terminal Help
client.py server.py x
TCP Socket > server.py > server_program
1 import hashlib
2 import socket
3
4 def server_program():
5     host = socket.gethostname()
6     port = 12202
7     server_socket = socket.socket()
8     server_socket.bind((host, port))
9     server_socket.listen(1)
10    conn, address = server_socket.accept()
11    key = "TranThiThuPhuong_B21DCAT151"
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("Nhan tu clien: " + str(data))
16    print("Nhan ma hoa SHA-512 của thông điệp trên từ Client: " + str(data_hash))
17
18    svhashed = hashlib.sha512(data.encode("utf-16") + key.encode("utf-16")).hexdigest()
19    data_s = 'Mã hóa thành công! \nHello, I am TranThiThuPhuong_B21DCAT151_server'
20    print('Server phản hồi tới Client: ' + data_s)
21    print("Server mã hóa SHA-512 của bạn rõ thông điệp thành: " + svhashed)
22
23    if data_hash != svhashed:
24        data_s = "The received message has lost its integrity."
25
26    conn.send(data_s.encode())
27    conn.close()
28
29 if __name__ == '__main__':
30     print("Server sẵn sàng!")
31     server_program()
```

Code của Server

Bài 15: Lập trình Client/Server để trao đổi thông tin an toàn

```
10     server_socket.accept()
11     key = "TranThiThuPhuong_B21DCAT151"
12
...
(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
$ /usr/bin/env /bin/python /home/tranphuong/.vscode/extensions/ms-python.debugpy-2024.4.0-linux-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 41251 -- /home/tranphuong/Downloads/ai/TCP\ Socket/server.py
Server san sang!
Nhan tu clien: Hello, I am TranThiThuPhuong_B21DCAT151_client
Nhan ma hoa SHA-512 cua thong diep tren tu Client: 7031258164fed84c761f4d2e1c40f40725819dc85b1c78a5f97c1cfc4f363ba83241ad5b04d7b367dbf23c85cb919828ea2cb33029d872321292c83200c494db
Server phan hoi toi Client: Ma hoa thanh cong!
Hello, I am TranThiThuPhuong_B21DCAT151_server
Server ma hoa SHA-512 cua ban ro thong diep thanh: 7031258164fed84c761f4d2e1c40f40725819dc85b1c78a5f97c1cfc4f363ba83241ad5b04d7b367dbf23c85cb919828ea2cb33029d872321292c83200c494db
(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
$ date
Thu Apr 25 01:17:29 EDT 2024
(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
```

Kết quả khi chạy



Code Client

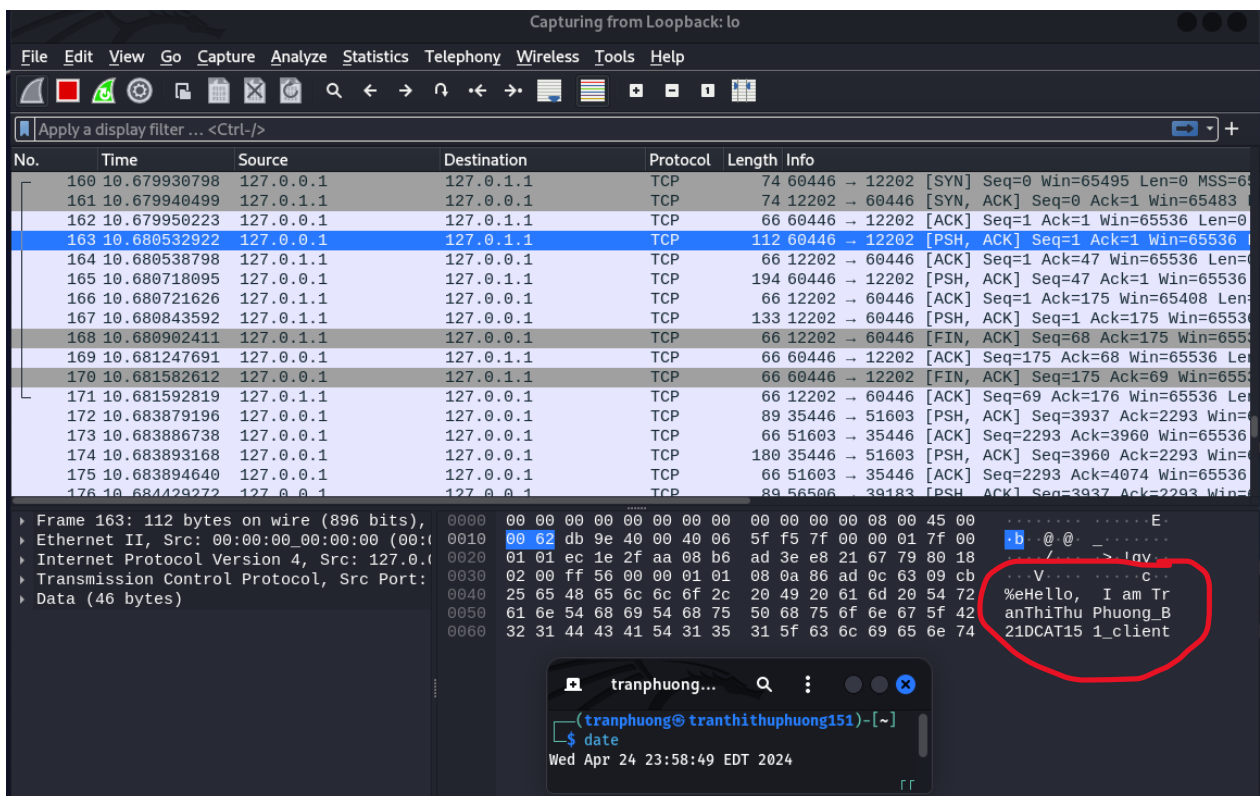
Bài 15: Lập trình Client/Server để trao đổi thông tin an toàn

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console + - [] [X] ... ^ X
(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
$ /usr/bin/env /bin/python /home/tranphuong/.vscode/extensions/ms-python.debugpy-2024.4.0-linux-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 39889 -- /home/tranphuong/Downloads/ai/TCP\ Socket/client.py
Client gui toi server: Hello, I am TranThiThuPhuong_B21DCAT151_client
Client gui ma hoa SHA512 toi Sever:7031258164fed84c761f4d2e1c40f40725819dc85b1c78a5f97c1cfc4f363ba83241ad5b04d7b367dbf23c85cb919828ea2cb33029d872321292c83200c494db
Nhan tu server: Ma hoa thanh cong!
Hello, I am TranThiThuPhuong_B21DCAT151_server

(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
$ date
Thu Apr 25 01:20:04 EDT 2024
```

Kết quả khi chạy Client

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



Client gửi thông điệp bản rõ đến Server

The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for file operations, capture control, and analysis. The main window is divided into three panes:

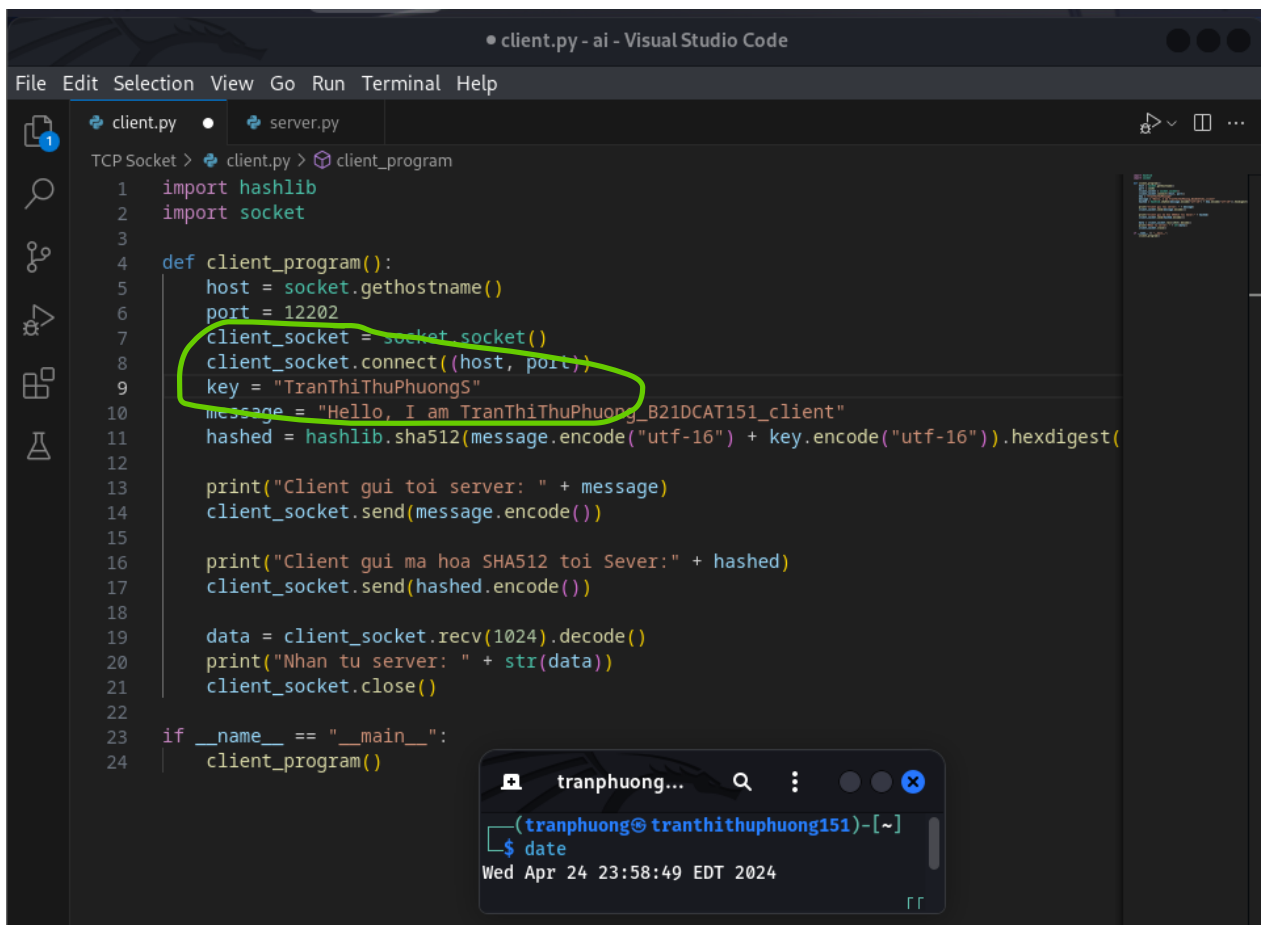
- Packet List:** Shows a list of captured packets. Packet 165 is selected and highlighted in blue. It is a TCP Reset (RST) packet with sequence number 16518095 and destination IP 127.0.0.1.
- Packet Details:** Shows the hierarchical structure of the selected packet. The layers are Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The TCP layer shows the reset flag set and the sequence number 16518095.
- Packet Bytes:** Shows the raw data of the selected packet in hexadecimal and ASCII. A red circle highlights the TCP sequence number field (0x00000000) in the hexadecimal view.

The status bar at the bottom indicates the current time and date: Wed Apr 24 23:58:49 EDT 2024.

[illegible]

- 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.”

Bài 15: Lập trình Client/Server để trao đổi thông tin an toàn

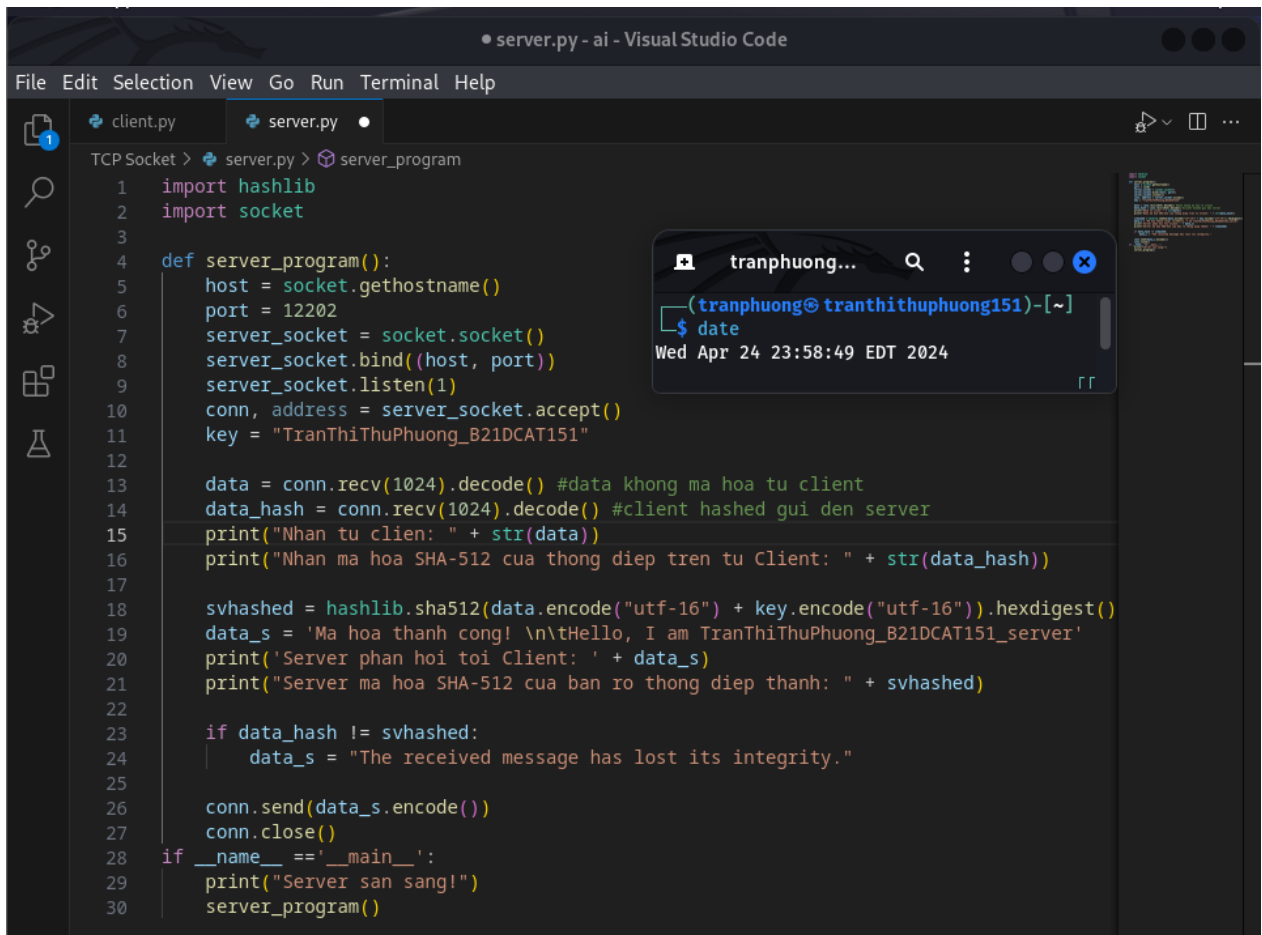


The screenshot shows a Visual Studio Code window with a file named `client.py` open. The code is a Python client program that connects to a server and sends a message and its SHA512 hash. A green circle highlights the `key` variable, which is set to `"TranThiThuPhuongS"`. Below the code, a terminal window is open, showing the command `date` being executed, which returns the current date and time: `Wed Apr 24 23:58:49 EDT 2024`.

```
client.py
1 import hashlib
2 import socket
3
4 def client_program():
5     host = socket.gethostname()
6     port = 12202
7     client_socket = socket.socket()
8     client_socket.connect((host, port))
9     key = "TranThiThuPhuongS"
10    message = "Hello, I am TranThiThuPhuong_B21DCAT151_client"
11    hashed = hashlib.sha512(message.encode("utf-16") + key.encode("utf-16")).hexdigest()
12
13    print("Client gui toi server: " + message)
14    client_socket.send(message.encode())
15
16    print("Client gui ma hoa SHA512 toi Sever:" + hashed)
17    client_socket.send(hashed.encode())
18
19    data = client_socket.recv(1024).decode()
20    print("Nhan tu server: " + str(data))
21    client_socket.close()
22
23 if __name__ == "__main__":
24     client_program()
```

Thay đổi key ở Client → Thông điệp mã hash thay đổi

Bài 15: Lập trình Client/Server để trao đổi thông tin an toàn



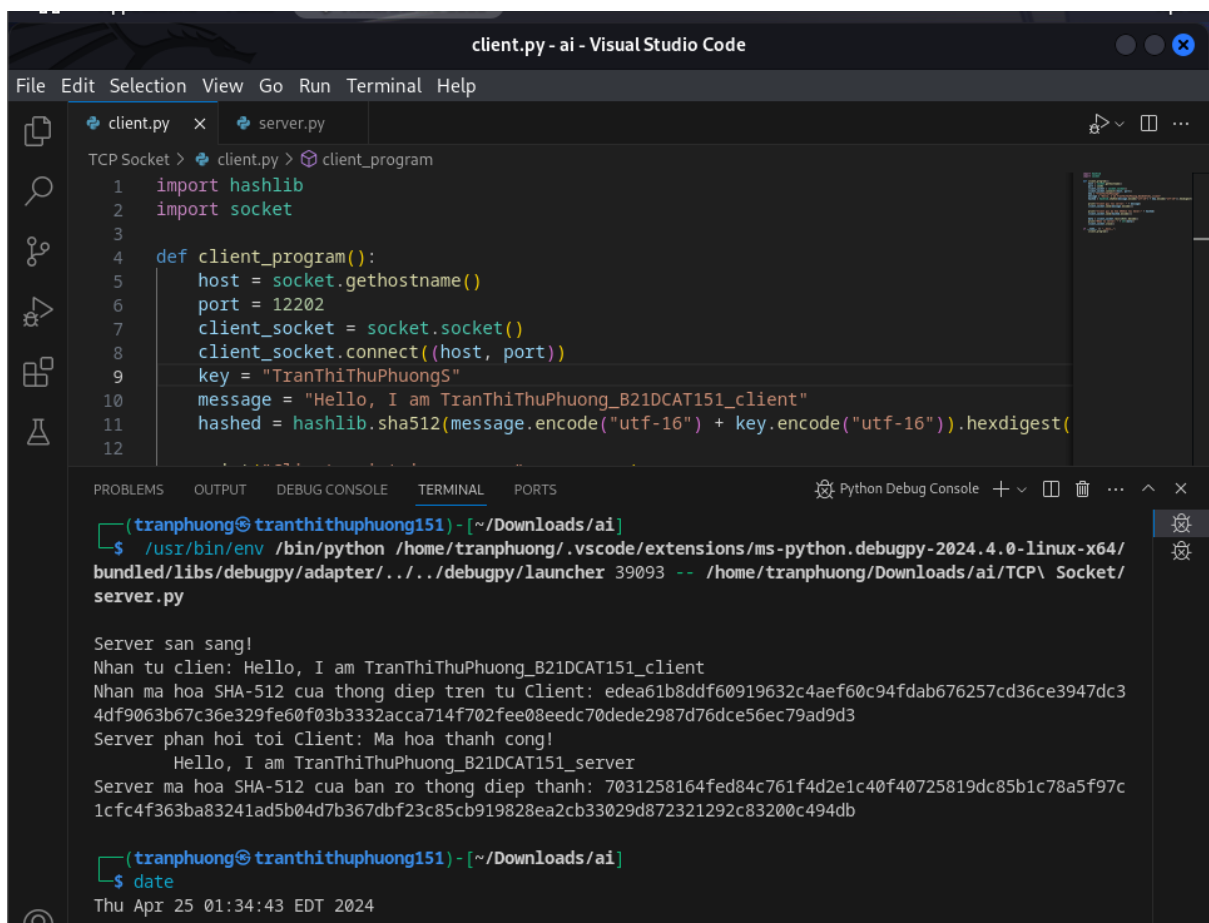
The screenshot shows a Visual Studio Code editor with a file named `server.py` open. The code is a Python script for a TCP socket server. It imports `hashlib` and `socket`. The `server_program()` function binds to `localhost` on port `12202`, listens for connections, and accepts a client. It receives data and a hash from the client, prints them, and then calculates a SHA-512 hash of the data combined with a key. It prints the result and sends a response back to the client. If the received hash does not match the calculated one, it prints an error message. The script is run as the main program.

```
1 import hashlib
2 import socket
3
4 def server_program():
5     host = socket.gethostname()
6     port = 12202
7     server_socket = socket.socket()
8     server_socket.bind((host, port))
9     server_socket.listen(1)
10    conn, address = server_socket.accept()
11    key = "TranThiThuPhuong_B21DCAT151"
12
13    data = conn.recv(1024).decode() #data khong ma hoa tu client
14    data_hash = conn.recv(1024).decode() #client hashed gui den server
15    print("Nhan tu clien: " + str(data))
16    print("Nhan ma hoa SHA-512 cua thong diep tren tu Client: " + str(data_hash))
17
18    svhashed = hashlib.sha512(data.encode("utf-16") + key.encode("utf-16")).hexdigest()
19    data_s = 'Ma hoa thanh cong! \nHello, I am TranThiThuPhuong_B21DCAT151_server'
20    print('Server phan hoi toi Client: ' + data_s)
21    print("Server ma hoa SHA-512 cua ban ro thong diep thanh: " + svhashed)
22
23    if data_hash != svhashed:
24        data_s = "The received message has lost its integrity."
25
26    conn.send(data_s.encode())
27    conn.close()
28
29 if __name__ == '__main__':
30     print("Server san sang!")
31     server_program()
```

A terminal window is open in the foreground, showing the prompt `tranphuong@tranthithuphuong151-[]` and the command `date`. The output of the command is `Wed Apr 24 23:58:49 EDT 2024`.

Bên Server giữ nguyên

Bài 15: Lập trình Client/Server để trao đổi thông tin an toàn



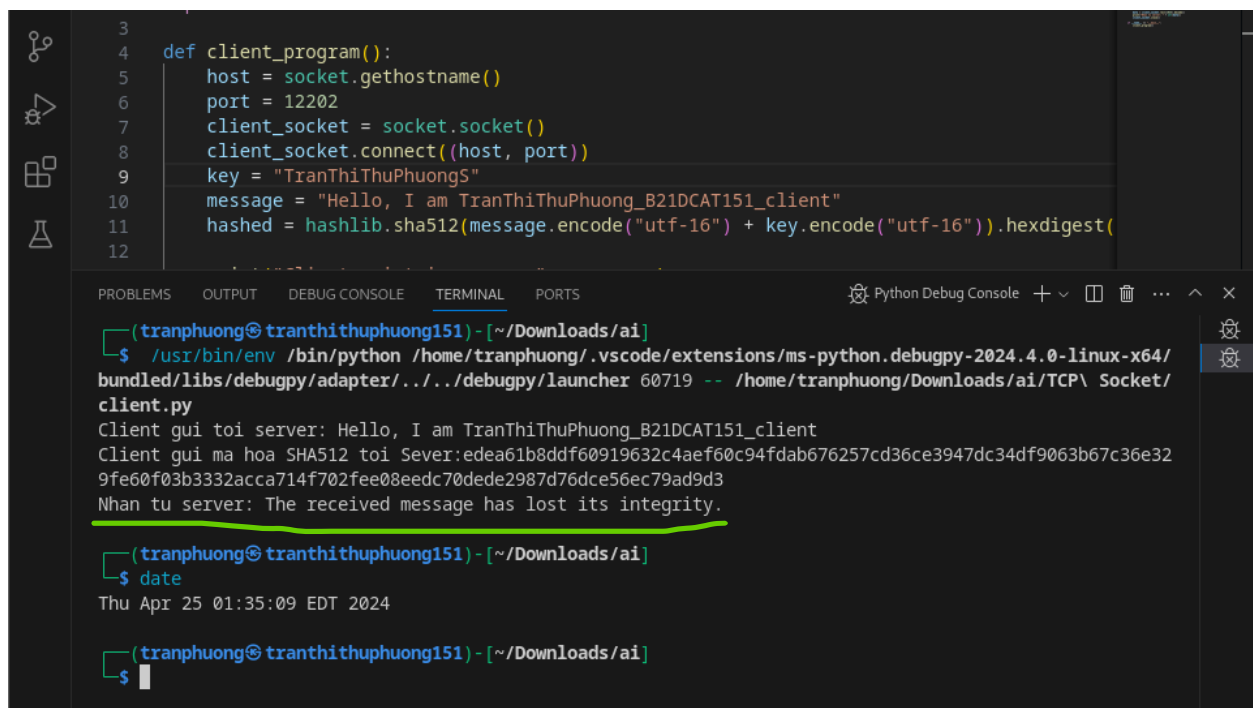
```
client.py - ai - Visual Studio Code
File Edit Selection View Go Run Terminal Help

client.py x server.py
TCP Socket > client.py > client_program
1 import hashlib
2 import socket
3
4 def client_program():
5     host = socket.gethostname()
6     port = 12202
7     client_socket = socket.socket()
8     client_socket.connect((host, port))
9     key = "TranThiThuPhuongS"
10    message = "Hello, I am TranThiThuPhuong_B21DCAT151_client"
11    hashed = hashlib.sha512(message.encode("utf-16") + key.encode("utf-16")).hexdigest()
12
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console
(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
$ /usr/bin/env /bin/python /home/tranphuong/.vscode/extensions/ms-python.debugpy-2024.4.0-linux-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 39093 -- /home/tranphuong/Downloads/ai/TCP\ Socket/server.py

Server san sang!
Nhan tu clien: Hello, I am TranThiThuPhuong_B21DCAT151_client
Nhan ma hoa SHA-512 cua thong diep tren tu Client: edea61b8ddf60919632c4aef60c94fdab676257cd36ce3947dc34df9063b67c36e329fe60f03b3332acca714f702fee08eedc70dede2987d76dce56ec79ad9d3
Server phan hoi toi Client: Ma hoa thanh cong!
Hello, I am TranThiThuPhuong_B21DCAT151_server
Server ma hoa SHA-512 cua ban ro thong diep thanh: 7031258164fed84c761f4d2e1c40f40725819dc85b1c78a5f97c1cfc4f363ba83241ad5b04d7b367dbf23c85cb919828ea2cb33029d872321292c83200c494db

(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
$ date
Thu Apr 25 01:34:43 EDT 2024
```

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



```
3
4 def client_program():
5     host = socket.gethostname()
6     port = 12202
7     client_socket = socket.socket()
8     client_socket.connect((host, port))
9     key = "TranThiThuPhuongS"
10    message = "Hello, I am TranThiThuPhuong_B21DCAT151_client"
11    hashed = hashlib.sha512(message.encode("utf-16") + key.encode("utf-16")).hexdigest()
12
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console
(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
$ /usr/bin/env /bin/python /home/tranphuong/.vscode/extensions/ms-python.debugpy-2024.4.0-linux-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 60719 -- /home/tranphuong/Downloads/ai/TCP\ Socket/client.py

Client gui toi server: Hello, I am TranThiThuPhuong_B21DCAT151_client
Client gui ma hoa SHA512 toi Sever: edea61b8ddf60919632c4aef60c94fdab676257cd36ce3947dc34df9063b67c36e329fe60f03b3332acca714f702fee08eedc70dede2987d76dce56ec79ad9d3
Nhan tu server: The received message has lost its integrity.

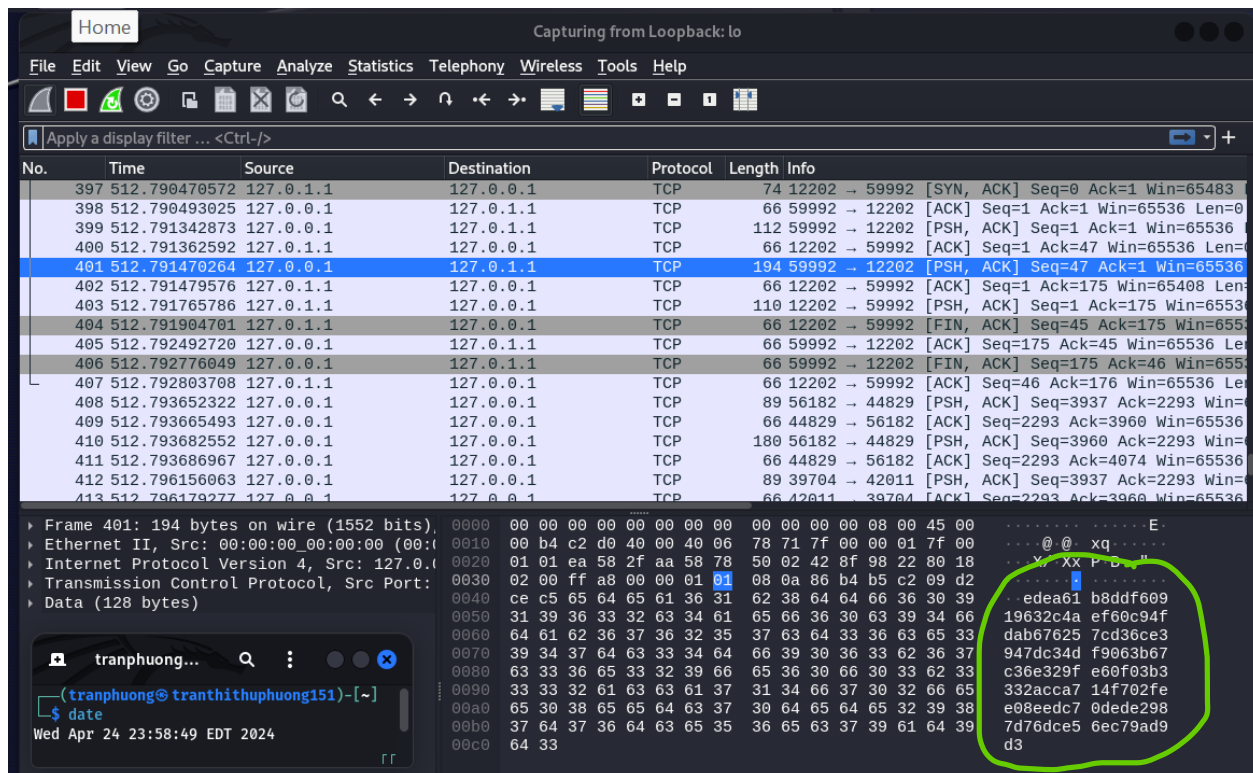
(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
$ date
Thu Apr 25 01:35:09 EDT 2024

(tranphuong@tranthithuphuong151) - [~/Downloads/ai]
$
```

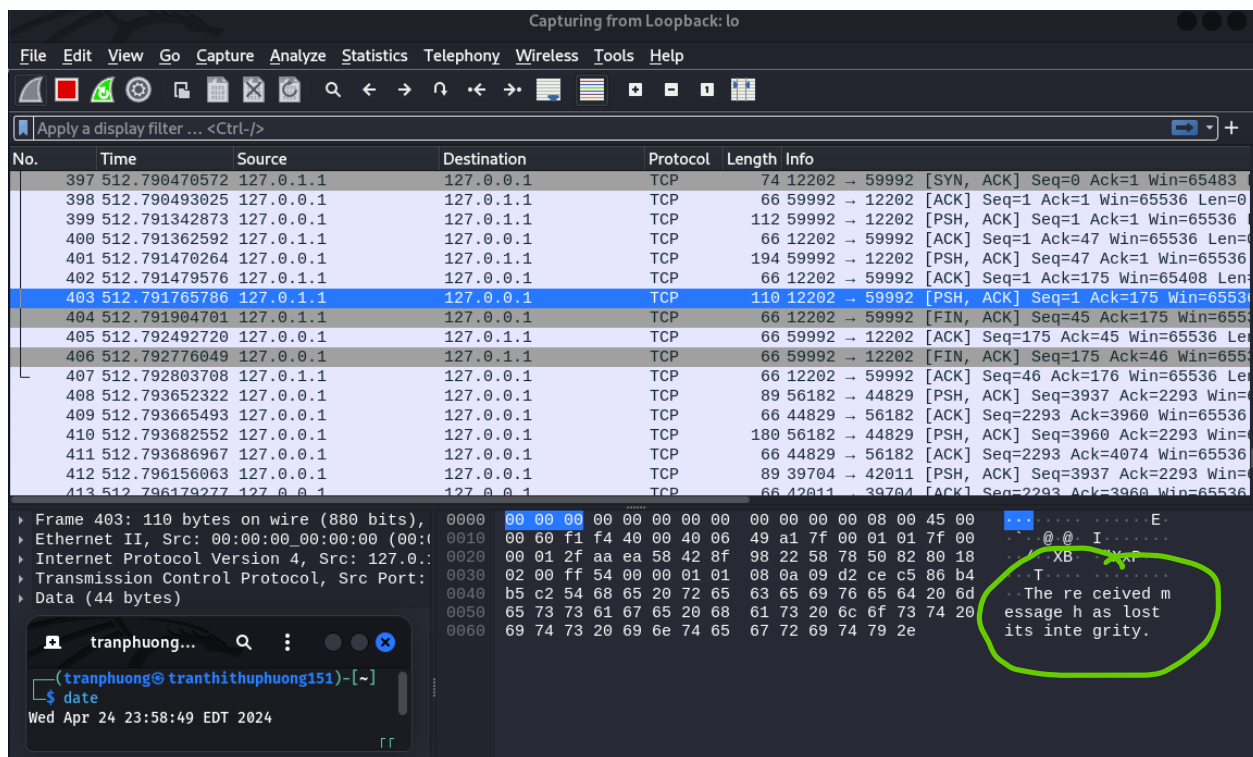
Kết quả, bên Client nhận được thông báo rằng dữ liệu đã bị mất mát

Bài 15: Lập trình Client/Server để trao đổi thông tin an toàn

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



Khi truyền từ Client → Server, Mã hash đã bị thay đổi



Server phản hồi Client: dữ liệu đã mất mát

3. Kết luận

- Lập trình và chạy thành công TCP Socket Client/Server
- Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi
- Bắt được các bản tin trao đổi giữa client và server trong Wireshark

4. 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](#)