

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



MÔN HỌC: THỰC TẬP CƠ SỞ
BÁO CÁO BÀI THỰC HÀNH SỐ 15

Giảng viên hướng dẫn : PGS.TS Hoàng Xuân Dậu
Sinh viên thực hiện : Nguyễn Nhật Minh
Mã sinh viên : B21DCAT132

Hà Nội, tháng 3 năm 2024

Môn học: Thực tập cơ sở

Bài 15: Lập trình client/server để trao đổi thông tin an toàn

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. Tìm hiểu lí thuyết

2.1. Socket là gì.

Lập trình mạng là một trong những nhiệm vụ căn bản để phát triển các ứng dụng. Một chương trình mạng được viết ra để các chương trình trên các máy tính khác nhau có thể truyền tin với nhau một cách hiệu quả và an toàn cho dù chúng được cài đặt trên mạng LAN, WAN hay mạng toàn cầu Internet, đây là điều căn bản đối với sự thành công của nhiều hệ thống.

Socket là một giao diện lập trình (API – Application Program Interface) ứng dụng mạng thông qua giao diện này có thể lập trình điều khiển việc truyền thông giữa 2 máy sử dụng các giao thức mức thấp như TCP,UDP... , Có thể tưởng tượng nó như một thiết bị truyền thông 2 chiều tương tự như tệp tin, chúng ta gửi/nhận dữ liệu giữa 2 máy, tương tự như việc đọc ghi trên tệp tin.

Đây chính là điểm cuối end-point tại liên kết truyền thông 2 chiều (two-way communication) và biểu diễn kết nối giữa Server - Client. Những lớp Socket hiện đang ràng buộc với 1 cổng port (thể hiện là 1 con số cụ thể) để những tầng TCP (hay TCP Layer) hoàn toàn có thể định danh được ứng dụng mà dữ liệu gửi đến.

2.2. Cơ chế hoạt động của Socket

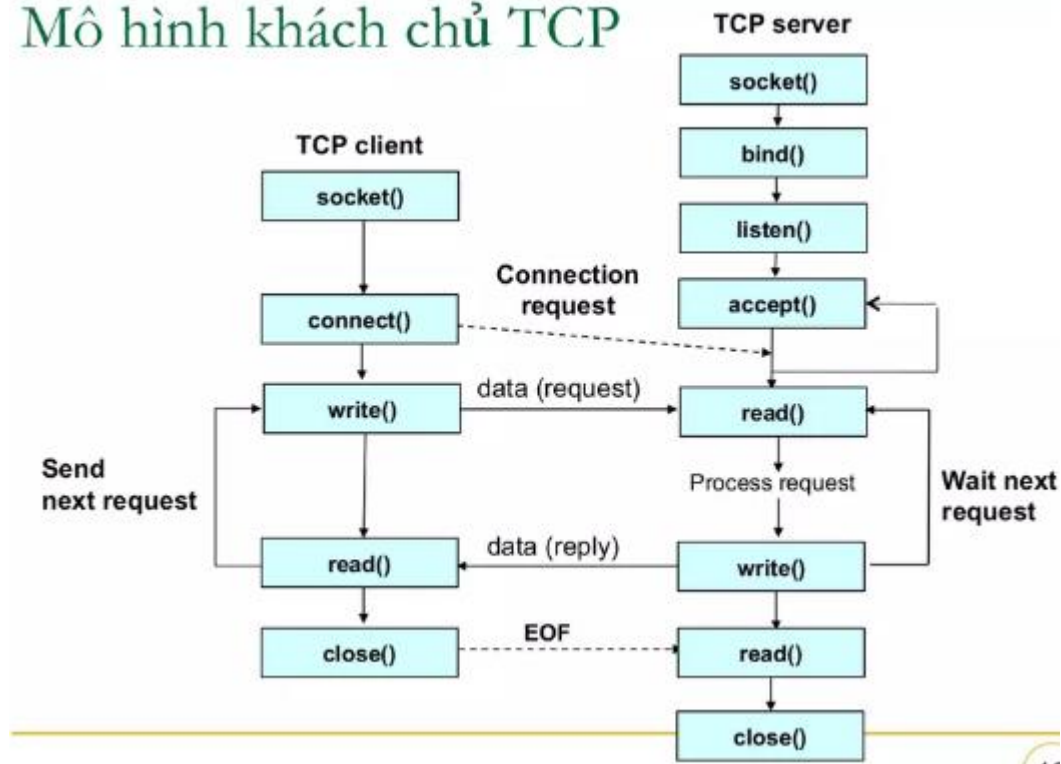
Hiện tại, chức năng của socket chính là kết nối giữa server và client thông qua UDP, TCP/IP để có thể truyền cũng như nhận nhận dữ liệu thông qua internet.

Hiện tại giao diện của lập trình ứng dụng mạng chỉ có thể hoạt động nếu như đã có những thông tin liên quan tới thông số IP cũng như số hiệu cổng của hai ứng dụng cần phải trao đổi dữ liệu.

Như vậy hai ứng dụng đang cần truyền thông tin bắt buộc phải đáp ứng được những điều kiện cơ bản sau đây thì socket mới hoạt động, cụ thể:

- Hai ứng dụng hoàn toàn có thể nằm cùng trên một máy hay hai máy khác nhau.
- Đối với trường hợp nếu như hai ứng dụng cùng trên một máy thì hiệu số cổng bắt buộc không được trùng với nhau.

Mô hình khách chủ TCP



Có thể phân quá trình kết nối làm 4 giai đoạn:

Giai đoạn 1: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết. Server sẵn sàng phục vụ Client.socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.

- bind(): Server yêu cầu gán số hiệu cổng (port) cho socket.
- listen(): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.

Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.

- socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
- connect(): Client gửi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
- accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.

Giai đoạn 3: Trao đổi thông tin giữa Client và Server.

- Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh `read()` và ngừng cho đến khi có thông điệp yêu cầu (Request Message) từ client gửi đến.
- Server phân tích và thực thi yêu cầu. Kết quả sẽ được gửi về client bằng lệnh `write()`.
- Sau khi gửi yêu cầu bằng lệnh `write()`, client chờ nhận thông điệp kết quả

(ReplyMessage) từ server bằng lệnh read().

Giai đoạn 4: Kết thúc phiên làm việc.

- Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).
- Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

3. Chuẩn bị môi trường

- Máy kali chứa wireshark để bắt gói tin và lập trình client/server

4. Thực hành

4.1. Lập trình client/server với TCP socket

Client

```
import socket
ClientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

ClientSocket.connect(("127.0.0.1", 8088))

ClientMessage = "Day la nguyen nhật minh b21dcat132 client"
print("Sending to server:", ClientMessage)
ClientSocket.send(ClientMessage.encode())

ServerMessage = ClientSocket.recv(1024).decode()
print("From Server:", ServerMessage)
ClientSocket.close()
```

Server

```
import socket
ServerName = "127.0.0.1"
ServerPort = 8088

ServerSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ServerSocket.bind(("127.0.0.1", ServerPort))
ServerSocket.listen(1)

c, addr = ServerSocket.accept()

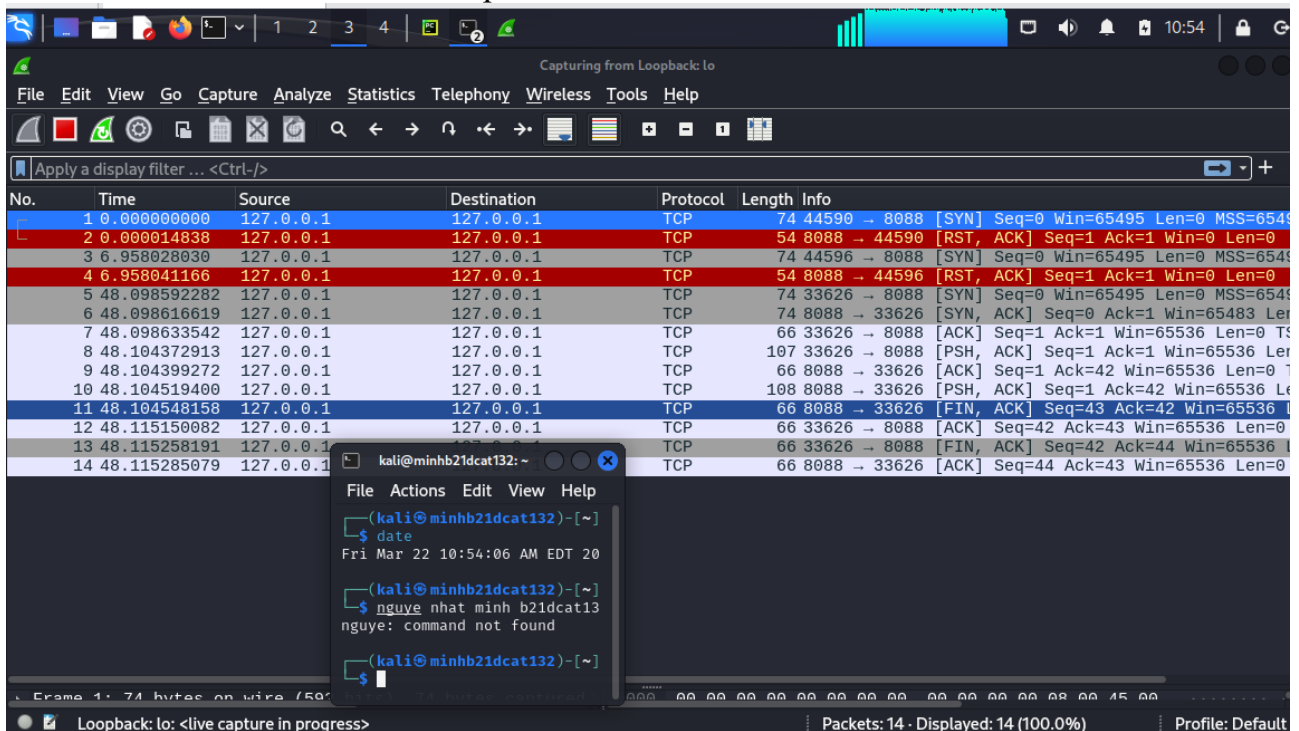
ClientMessage = c.recv(1024).decode()
print("From:", addr, ClientMessage)

ServerMessage = "This is nguyen nhật minh b21dcat132 server"
c.send(ServerMessage.encode())
c.close()
```

- Ta chạy và ra kết quả như hình

```
Run  server x client x
Sending to server: Day la nguyen nhat minh b21dcat132 client
From Server: This is nguyen nhat minh b21dcat132 server
Process finished with exit code 0
```

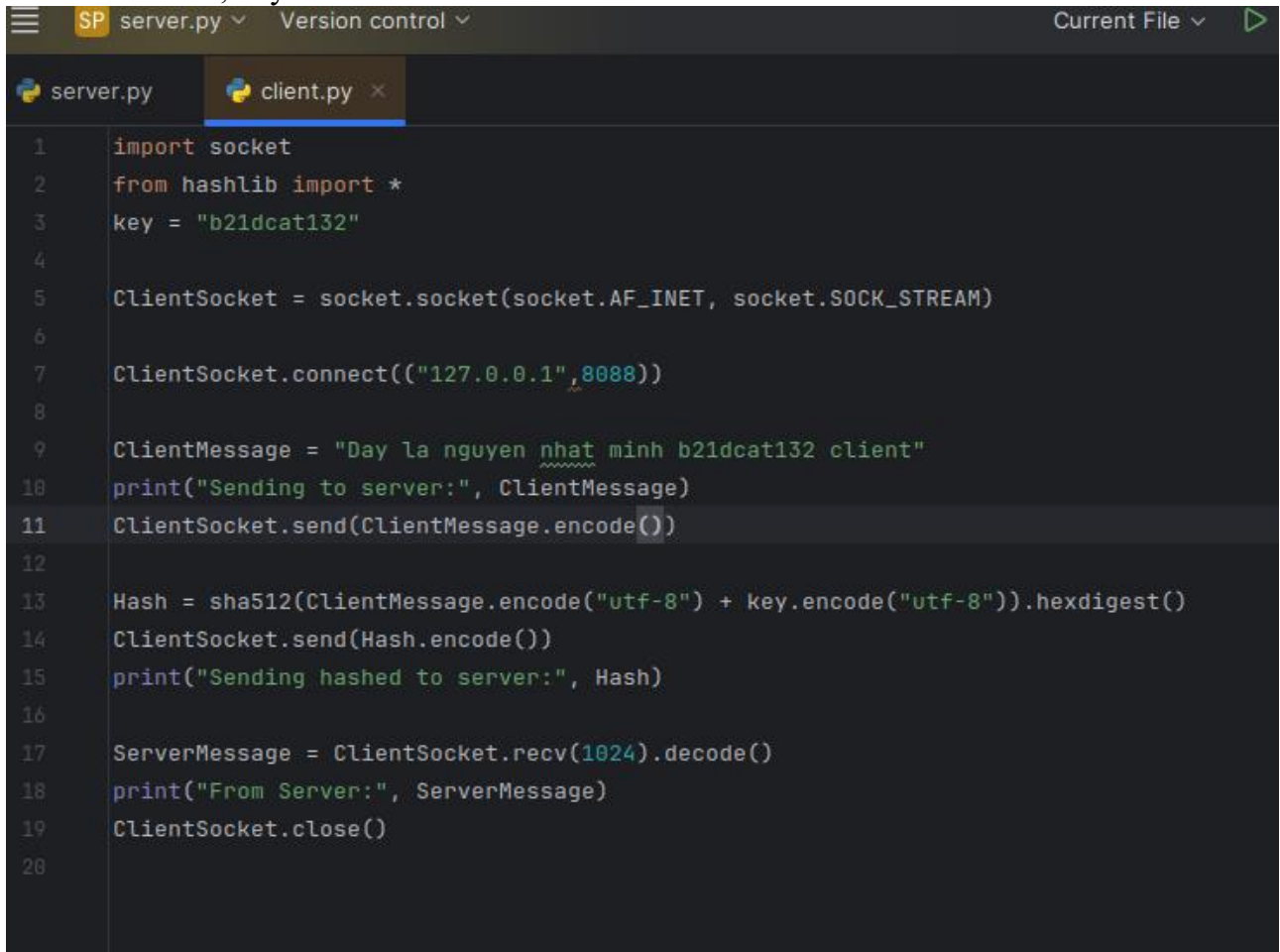
- Bật wireshark và bắt card loopback: lo



- Wireshark đã bắt thành công gói tin từ client tới server và ngược lại

4.2. Trao đổi thông điệp giữa Client và Server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi

Code của client, key là b21dcat132



```
1 import socket
2 from hashlib import *
3 key = "b21dcat132"
4
5 ClientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6
7 ClientSocket.connect(("127.0.0.1", 8088))
8
9 ClientMessage = "Day la nguyen nhat minh b21dcat132 client"
10 print("Sending to server:", ClientMessage)
11 ClientSocket.send(ClientMessage.encode())
12
13 Hash = sha512(ClientMessage.encode("utf-8") + key.encode("utf-8")).hexdigest()
14 ClientSocket.send(Hash.encode())
15 print("Sending hashed to server:", Hash)
16
17 ServerMessage = ClientSocket.recv(1024).decode()
18 print("From Server:", ServerMessage)
19 ClientSocket.close()
20
```

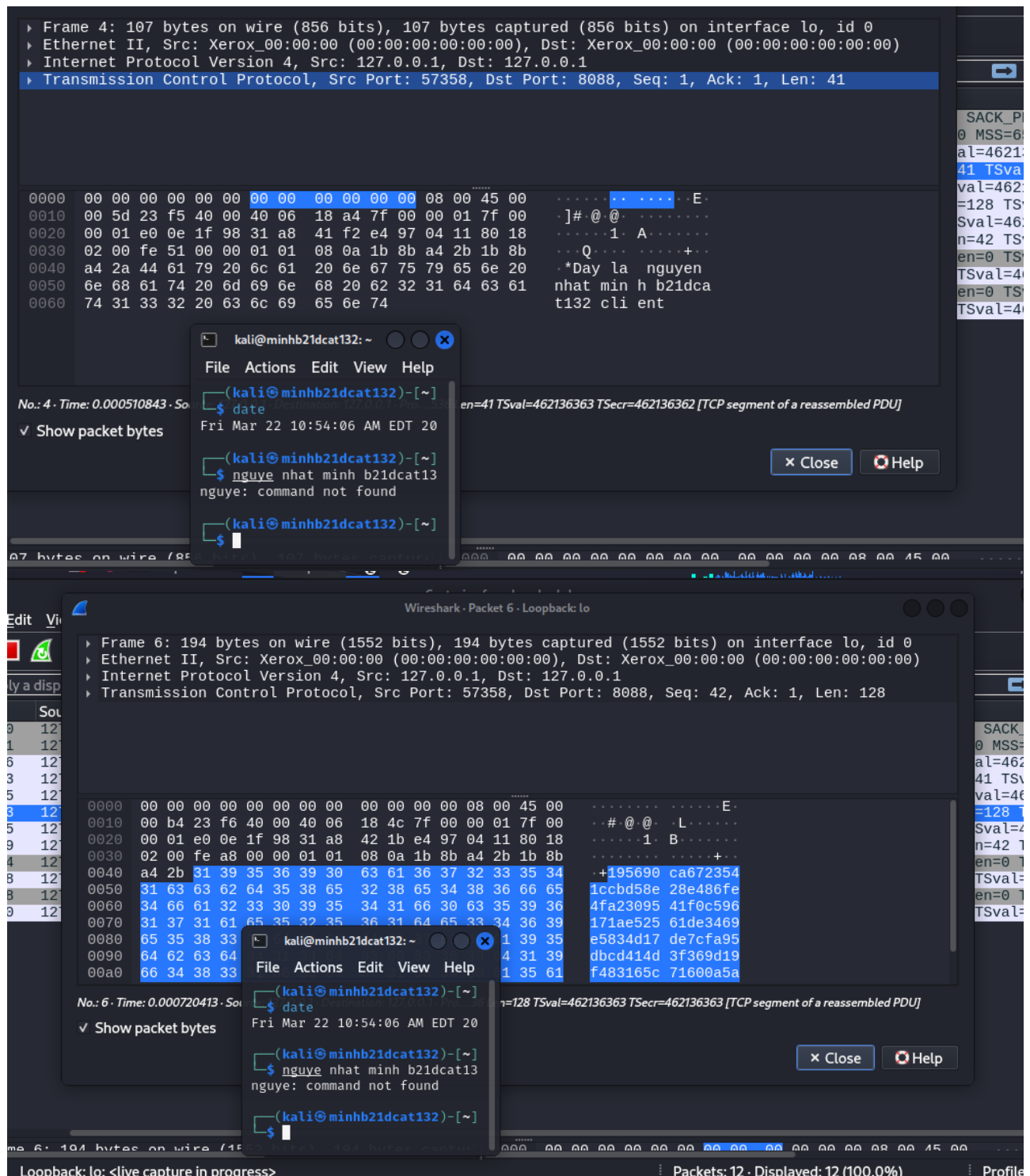

- Code của server

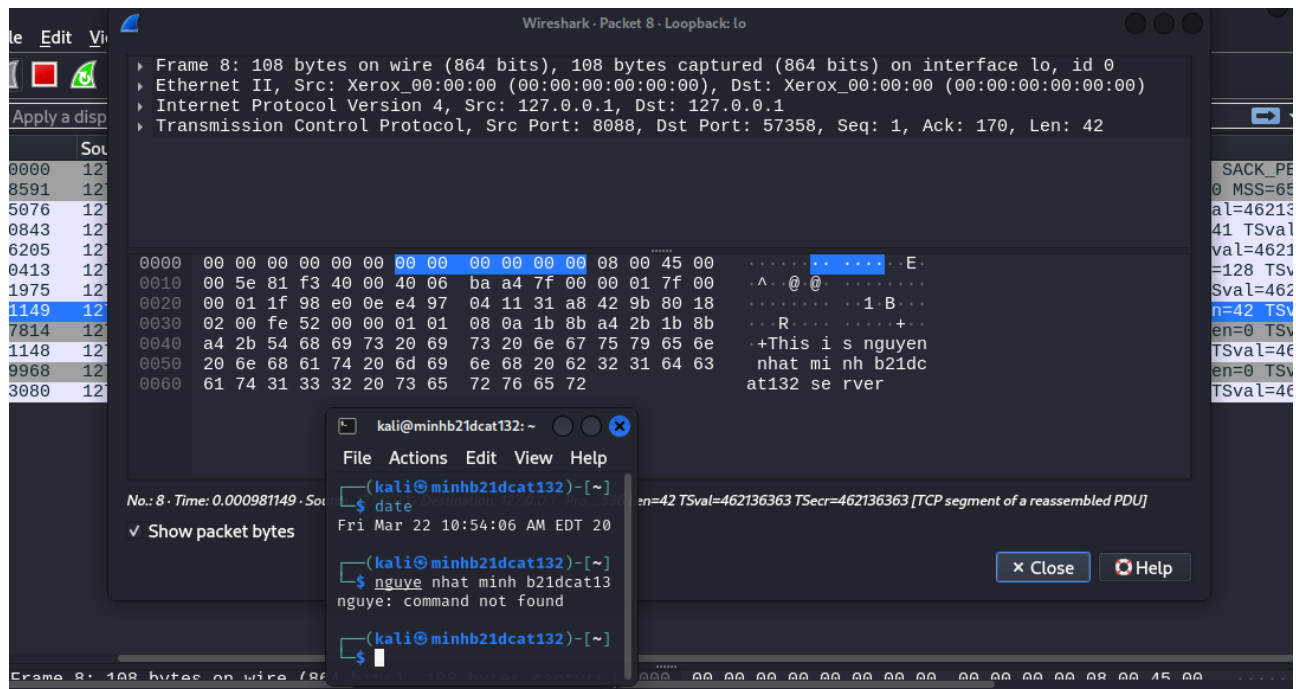
```
server.py x client.py
2  from hashlib import *
3  key = "b21dcat132"
4  ServerName = "127.0.0.1"
5  ServerPort = 8088
6  ServerSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7
8  ServerSocket.bind(("127.0.0.1", ServerPort))
9
10 ServerSocket.listen(1)
11
12 c, addr = ServerSocket.accept()
13
14 ClientMessage = c.recv(1024).decode()
15 HashClient = c.recv(1024).decode()
16 print("From:", addr, ClientMessage)
17 if HashClient != sha512(ClientMessage.encode("utf-8")+key.encode("utf-8")).hexdigest():
18     print("The received message has lost its integrity")
19     ServerMessage = "The received message has lost its integrity"
20     c.send(ServerMessage.encode())
21 else:
22     ServerMessage = "This is nguyen nhat minh b21dcat132 server"
23     c.send(ServerMessage.encode())
```

- Đây là kết quả khi chạy thành công và trao đổi toàn vẹn

```
PC SP server.py v Version control v Current File v
server.py x client.py x
2  from hashlib import *
3  key = "b21dcat132"
4
5  ClientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6
7  ClientSocket.connect(("127.0.0.1", 8088))
8
9  ClientMessage = "Day la nguyen nhat minh b21dcat132 client"
10 print("Sending to server:", ClientMessage)
11 ClientSocket.send(ClientMessage.encode())
12
13 Hash = sha512(ClientMessage.encode("utf-8") + key.encode("utf-8")).hexdigest()
14 ClientSocket.send(Hash.encode())

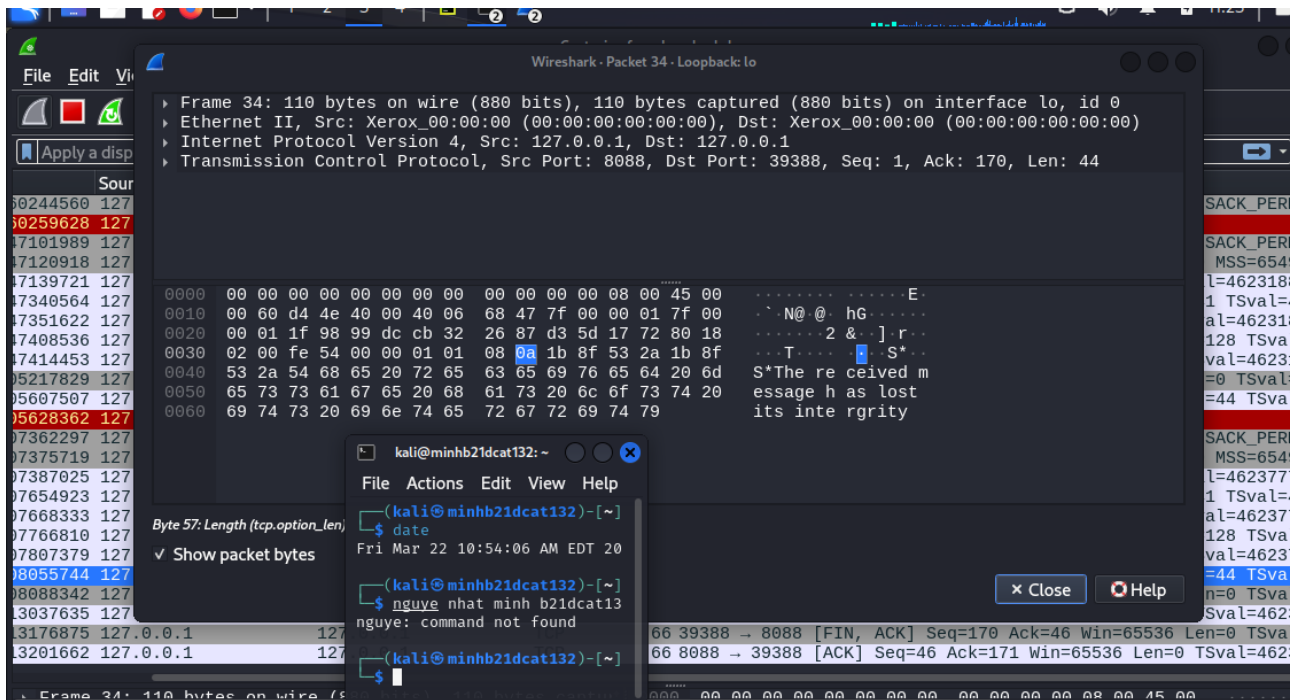
Run server x client x
/usr/bin/python3.11 /home/kali/client.py
Sending to server: Day la nguyen nh
```



- Nếu ta sửa đổi 2 key khác nhau thì giá trị toàn vẹn cũng sẽ mất





5. Kết luận

- Bài thực hành hoàn thành vào ngày 22/03/2024

