Lab

3

Phân tích hoạt động giao thức TCP - UDP

TCP/UDP Protocol

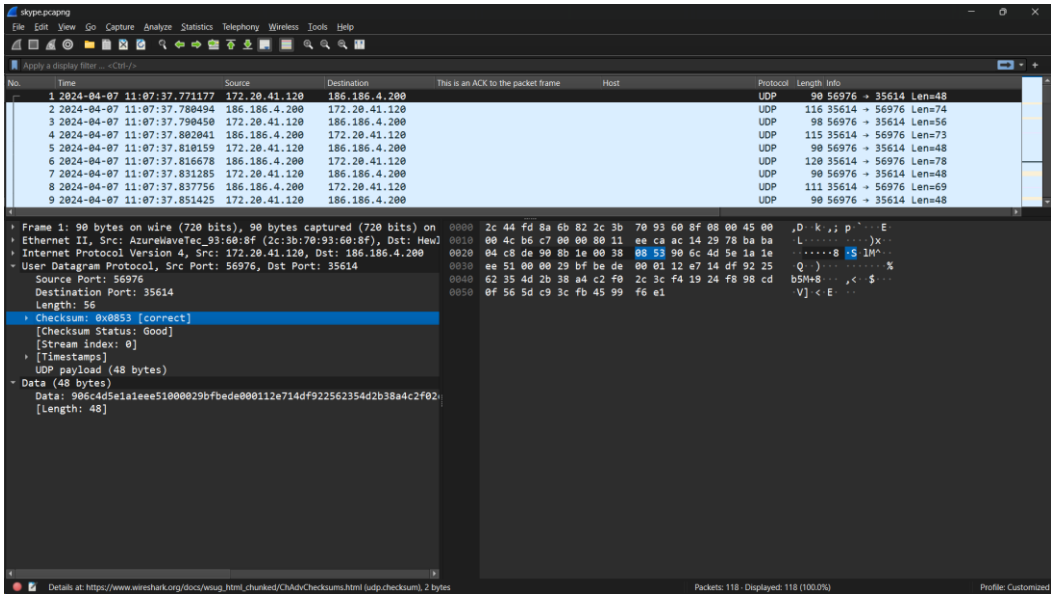
1. **Tính checksum của 1 gói tin UDP bất kỳ (dựa trên thông tin bắt được từ wireshark checksum**

Trả lời:

Pseudo header starts here			
	Decimal	Binary	Hex
Source IP	192.168 0.31	1100 0000 1010 1000 0000 0000 0001 1111	C0 A8 00 1F
Destination IP	192.168. 0.30	1100 0000 1010 1000 0000 0000 0001 1110	C0 A8 00 1E
Reserved/UDP protocol	0/17	0000 0000 0001 0001	00 11
Padding/Length	0/10	0000 0000 0000 1010	00 0A
Pseudo header ends here so we will add the real UDP header to this			
UDP Source Port	20	0000 0000 0001 0100	00 14
UDP destination Port	10	0000 0000 0000 1010	00 0A
UDP Length	10	0000 0000 0000 1010	00 0A
UDP Data	Hi	0100 1000 0110 1001	48 69
Now that we have all that information let's add			
		1 1100 1010 0011 1001	1 CA 39
Notice in our previous entry our values exceed 16 bits (2 bytes). This will not work since our checksum has to be 16 bits. To get to 16 bits we will expand the results from t to become 32 bits. Thus we will prepend hex 000 to 1 CA 39. We will also find the binary value of 000 and add it to the binary column.			
		1 1100 1010 0011 1001	00 01 CA 39
Now that we have the 32 Bit value we take the upper half 00 01 and add them to the lower half CA 39			
		0000 0000 0000 0001 + 1100 1010 0011 1001	00 01 + CA 39
		1100 1010 0011 1010	CA3A
We're getting there. Now that we have the above value, we need to find its one's complement. To do this we interchange the 0s and the 1s of the result above			
		0011 0101 1100 0101	**35 C5**

Ta dựa vào bảng, ta có thể thiết lập công thức.

```
PS C:\Users\DucAnh> python -u "c:\Users\DucAnh\OneDrive\Documents\Study\UIT\hk2\NMMT\TH\Lab 3\checksum.py"
Input Ip of Source: 172.20.41.120
Input Ip of Destination: 186.186.4.200
Input port of Source: 56976
Input port of Destination: 35614
Input length: 56
Input payload: 906c4d5e1a1eee51000029bfbede000112e714df922562354d2b38a4c2f02c3cf41924f898cd0f565dc93cfb4599f6e1
Check sum: 0x0853
```



2. Tìm gói tin tương ứng với các sự kiện sau:

TCP sender

- Sự kiện: Nhận dữ liệu từ tầng ứng dụng

- Sự kiện: timeout

- Sự kiện: nhận được ACK

TCP Receiver: ACK generation [RFC 5681]

- Nhận được segment đúng với STT đang chờ. Tất cả segment trước đó đã được ACK.

- Nhận được segment đúng với STT đang chờ, một segment chưa được ACK.

- Nhận được segment không đúng thứ tự (STT cao hơn).

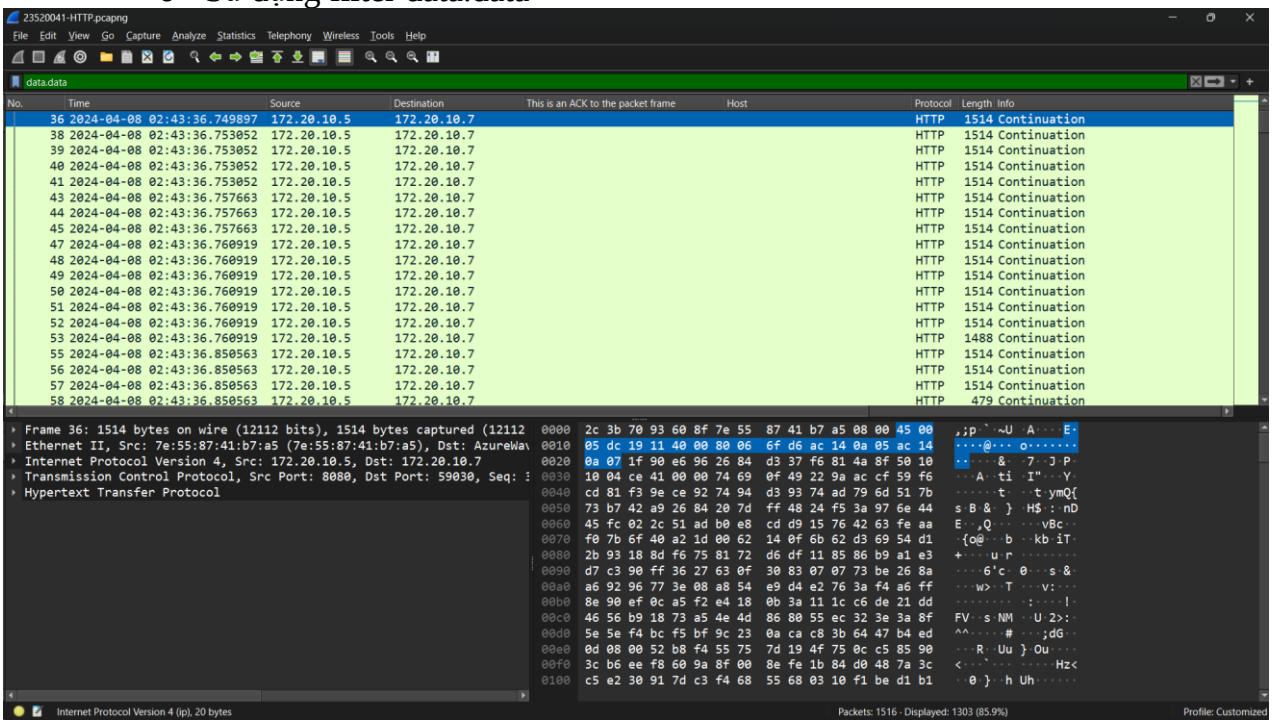
- Nhận được segment trong khoảng bị trống (giữa STT đang chờ và STT nhận được trước đó).

Trả lời:

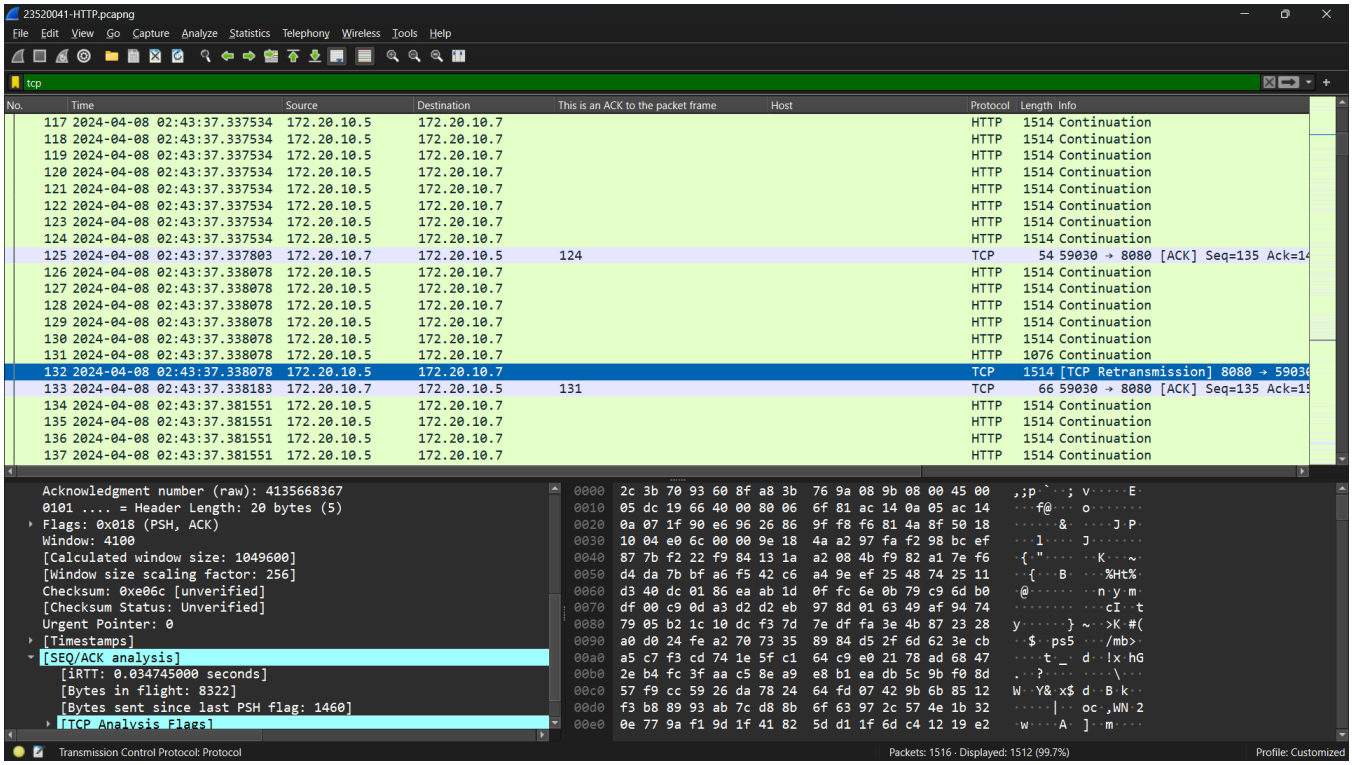
TCP sender

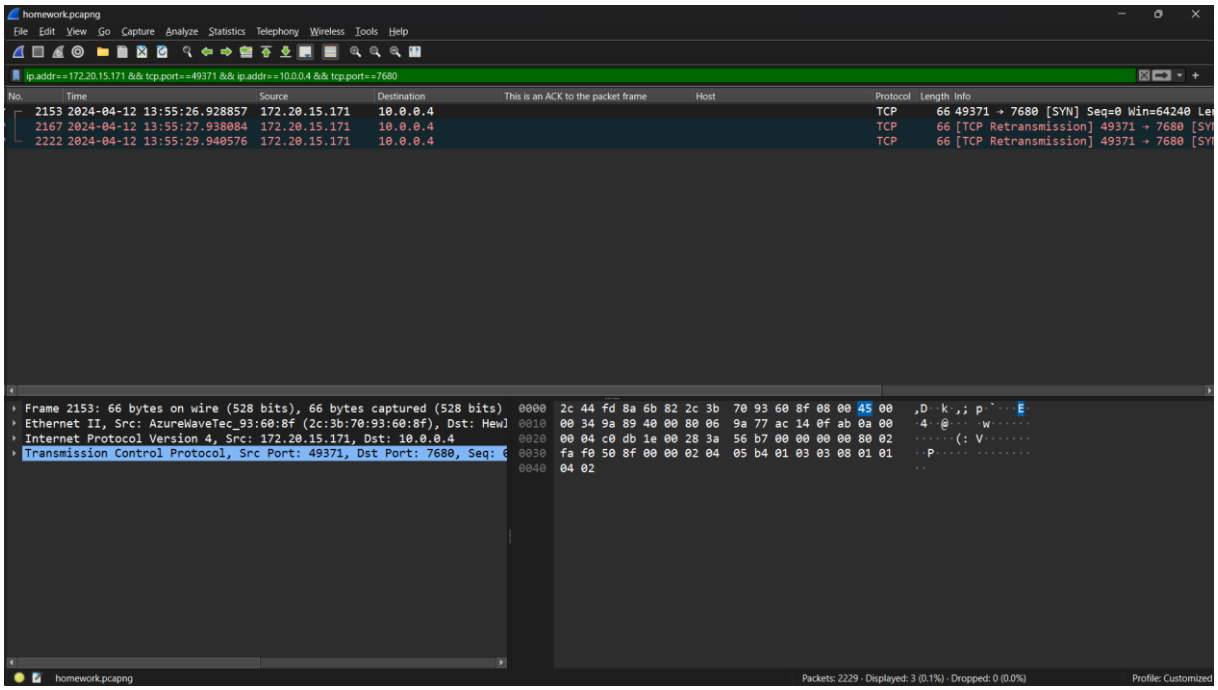
- ***Sự kiện: nhận dữ liệu từ tầng ứng dụng***

  - Sử dụng filter data.data



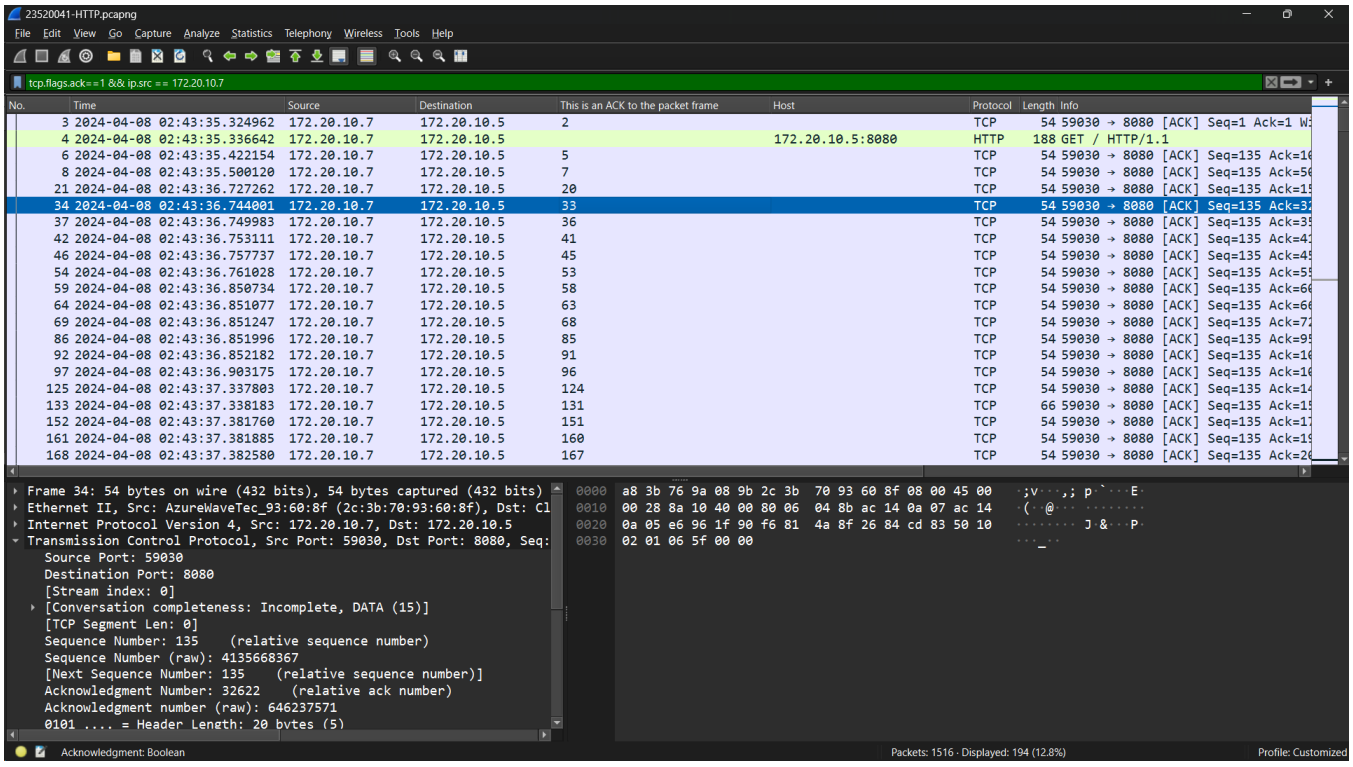
- ***Sự kiện: timeout***





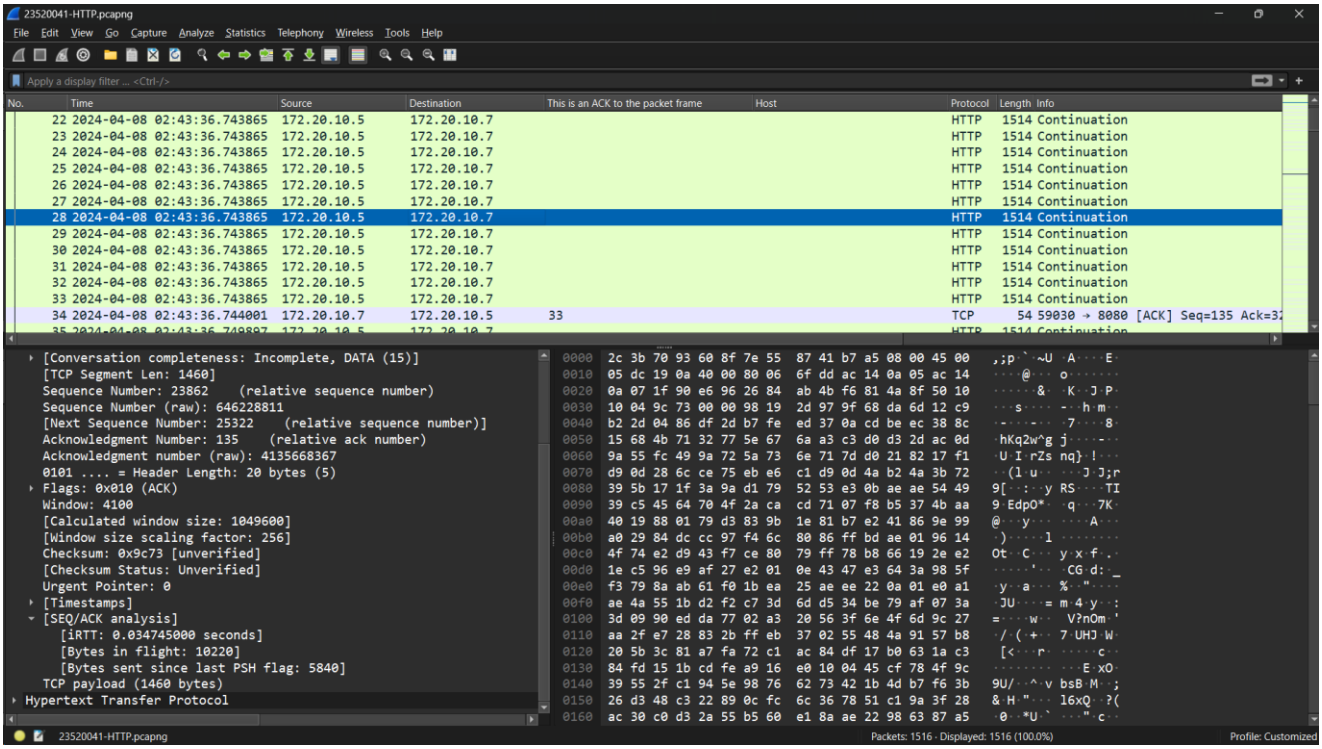
- ***Sự kiện: nhận được ACK***

  - Sử dụng filter: tcp.flags.ack==1 && ip.src == 172.20.10.7



TCP receiver

- Nhận được segment đúng với STT đang chờ. Tất cả segment trước đó đã được ACK.



- Nhận được segment đúng với STT đang chờ, một segment chưa được ACK.

Không có

- Nhận được segment không đúng thứ tự (STT cao hơn).

Không có

- Nhận được segment trong khoảng bị trống (giữa STT đang chờ và STT nhận được trước đó)

Không có