TCP Three-way Handshake Analysis - Step 1

Using the output of protocol analysis software, such as Wireshark outputs, you can examine the operation of the TCP 3-way handshake:

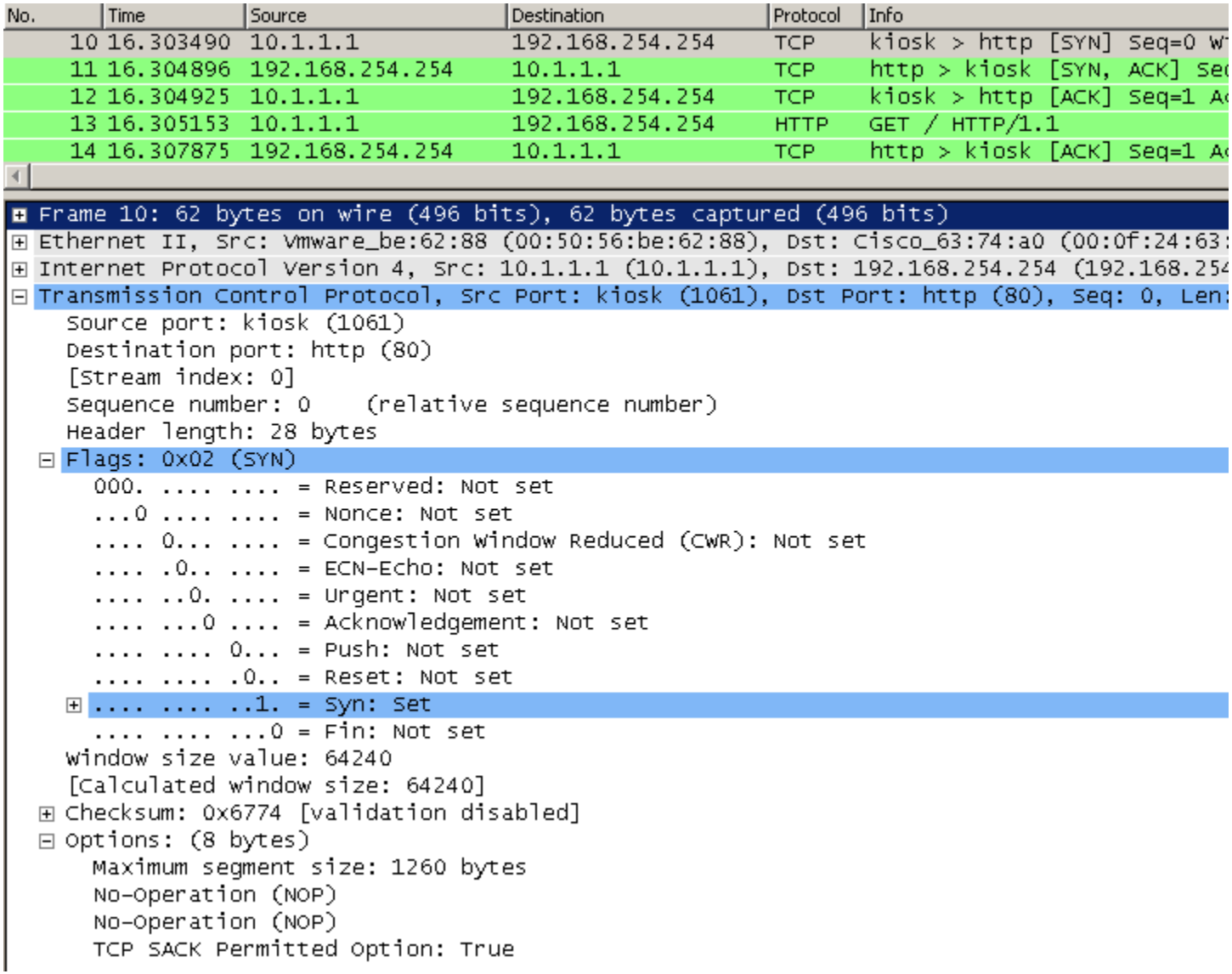
Step 1: The initiating client requests a client-to-server communication session with the server.

A TCP client begins the three-way handshake by sending a segment with the synchronize sequence number (SYN) control flag set, indicating an initial value in the sequence number field in the header. This initial value for the sequence number, known as the initial sequence number (ISN), is randomly chosen and is used to begin tracking the flow of data from the client to the server for this session. The ISN in the header of each segment is increased by one for each byte of data sent from the client to the server as the data conversation continues.

As shown in Figure 1, output from a protocol analyzer shows the SYN control flag and the relative sequence number.

The SYN control flag is set and the relative sequence number is at 0. Although the protocol analyzer in the graphic indicates the relative values for the sequence and acknowledgement numbers, the true values are 32-bit binary numbers. The figure shows the four bytes represented in hexadecimal.

Figure 1 - TCP 3-Way Handshake (SYN)



TCP Three-way Handshake Analysis - Step 2

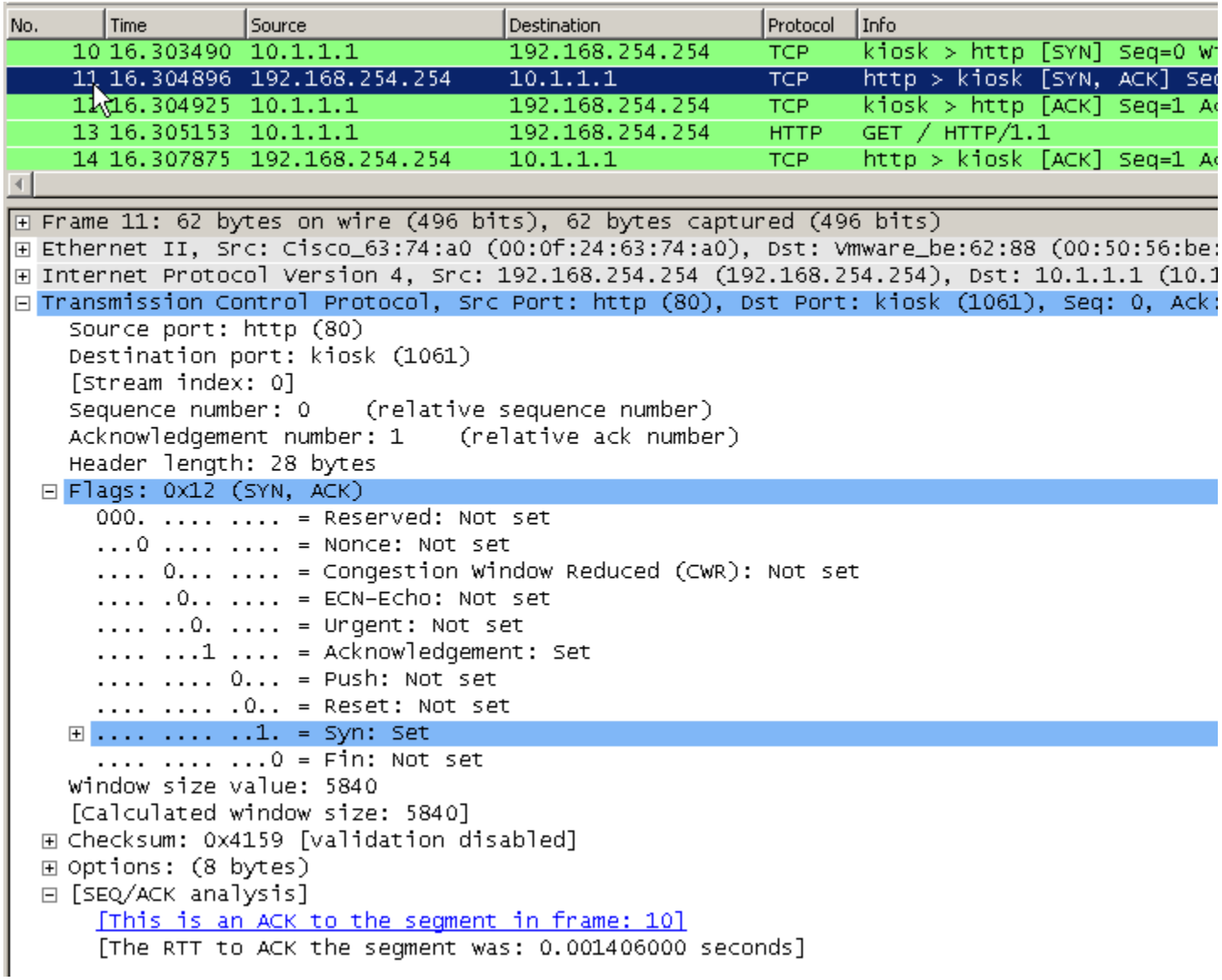
Step 2: The server acknowledges the client-to-server communication session and requests a server-to-client communication session.

The TCP server must acknowledge the receipt of the SYN segment from the client to establish the session from the client to the server. To do so, the server sends a segment back to the client with the acknowledgement (ACK) flag set indicating that the acknowledgment number is significant. With this flag set in the segment, the client recognizes this as an acknowledgement that the server received the SYN from the TCP client.

The value of the acknowledgment number field is equal to the ISN plus 1. This establishes a session from the client to the server. The ACK flag remains set for the balance of the session. Recall that the conversation between the client and the server is actually two one-way sessions: one from the client to the server, and the other from the server to the client. In this second step of the three-way handshake, the server must initiate the response to the client. To start this session, the server uses the SYN flag in the same way that the client did. It sets the SYN control flag in the header to establish a session from the server to the client. The SYN flag indicates that the initial value of the sequence number field is in the header. This value is used to track the flow of data in this session from the server back to the client.

As shown Figure 2, the protocol analyzer output shows that the ACK and SYN control flags are set and the relative sequence and acknowledgement numbers are displayed.

Figure 2 - TCP 3-Way Handshake (SYN, ACK)



TCP Three-way Handshake Analysis - Step 3

Step 3: The initiating client acknowledges the server-to-client communication session.

Finally, the TCP client responds with a segment containing an ACK that is the response to the TCP SYN sent by the server. There is no user data in this segment. The value in the acknowledgment number field contains one more than the ISN received from the server. After both sessions are established between client and server, all additional segments exchanged in this communication will have the ACK flag set.

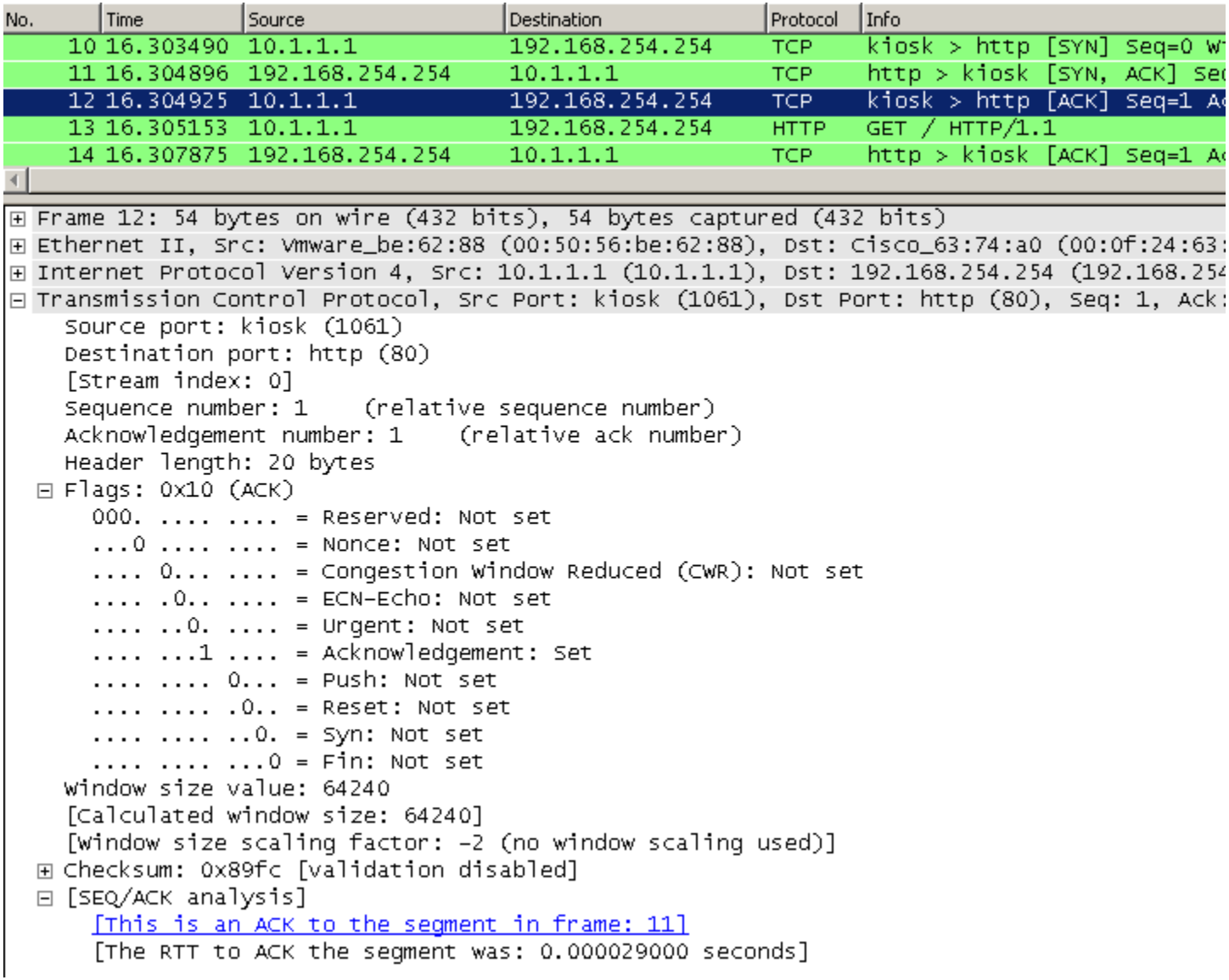
As shown in Figure 3, the protocol analyzer output shows the ACK control flag set and the relative sequence and acknowledgement numbers.

Security can be added to the data network by:

- Denying the establishment of TCP sessions
- Only allowing sessions to be established for specific services
- Only allowing traffic as a part of already established sessions

These security measures can be implemented for all TCP sessions or only for selected sessions.

Figure 3 - TCP 3-Way Handshake (ACK)



No.	Time	Source	Destination	Protocol	Info
10	16.303490	10.1.1.1	192.168.254.254	TCP	kiosk > http [SYN] Seq=0 W
11	16.304896	192.168.254.254	10.1.1.1	TCP	http > kiosk [SYN, ACK] Se
12	16.304925	10.1.1.1	192.168.254.254	TCP	kiosk > http [ACK] Seq=1 A
13	16.305153	10.1.1.1	192.168.254.254	HTTP	GET / HTTP/1.1
14	16.307875	192.168.254.254	10.1.1.1	TCP	http > kiosk [ACK] Seq=1 A

```
⊞ Frame 12: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
⊞ Ethernet II, Src: Vmware_be:62:88 (00:50:56:be:62:88), Dst: Cisco_63:74:a0 (00:0f:24:63:
⊞ Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1), Dst: 192.168.254.254 (192.168.254
⊟ Transmission Control Protocol, Src Port: kiosk (1061), Dst Port: http (80), Seq: 1, Ack:
    Source port: kiosk (1061)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x10 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgement: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
    Window size value: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0x89fc [validation disabled]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 11]
      [The RTT to ACK the segment was: 0.000029000 seconds]
```

TCP Session Termination Analysis

To close a connection, the Finish (FIN) control flag must be set in the segment header. To end each one-way TCP session, a two-way handshake is used, consisting of a FIN segment and an ACK segment. Therefore, to terminate a single conversation supported by TCP, four exchanges are needed to end both sessions, as shown in Figure 1.

Note: In this explanation, the terms client and server are used as a reference for simplicity, but the termination process can be initiated by any two hosts that have an open session:

Step 1: When the client has no more data to send in the stream, it sends a segment with the FIN flag set.

Step 2: The server sends an ACK to acknowledge the receipt of the FIN to terminate the session from client to server.

Step 3: The server sends a FIN to the client, to terminate the server to client session.

Step 4: The client responds with an ACK to acknowledge the FIN from the server.

When the client has no more data to transfer, it sets the FIN flag in the header of a segment. Next, the server end of the connection sends a normal segment containing data with the ACK flag set using the acknowledgment number, confirming that all the bytes of data have been received. When all segments have been acknowledged, the session is closed.

The session in the other direction is closed using the same process. The receiver indicates that there is no more data to send by setting the FIN flag in the header of a segment sent to the source. A return acknowledgement confirms that all bytes of data have been received and that session is, in turn, closed.

Refer to Figure 4 and 5 to see the FIN and ACK control flags set in the segment header, thereby closing a HTTP session.

It is also possible to terminate the connection by a three-way handshake. When the client has no more data to send, it sends a FIN to the server. If the server also has no more data to send, it can reply with both the FIN and ACK flags set, combining two steps into one. The client then replies with an ACK.

Figure 4 - TCP Session Termination (FIN)

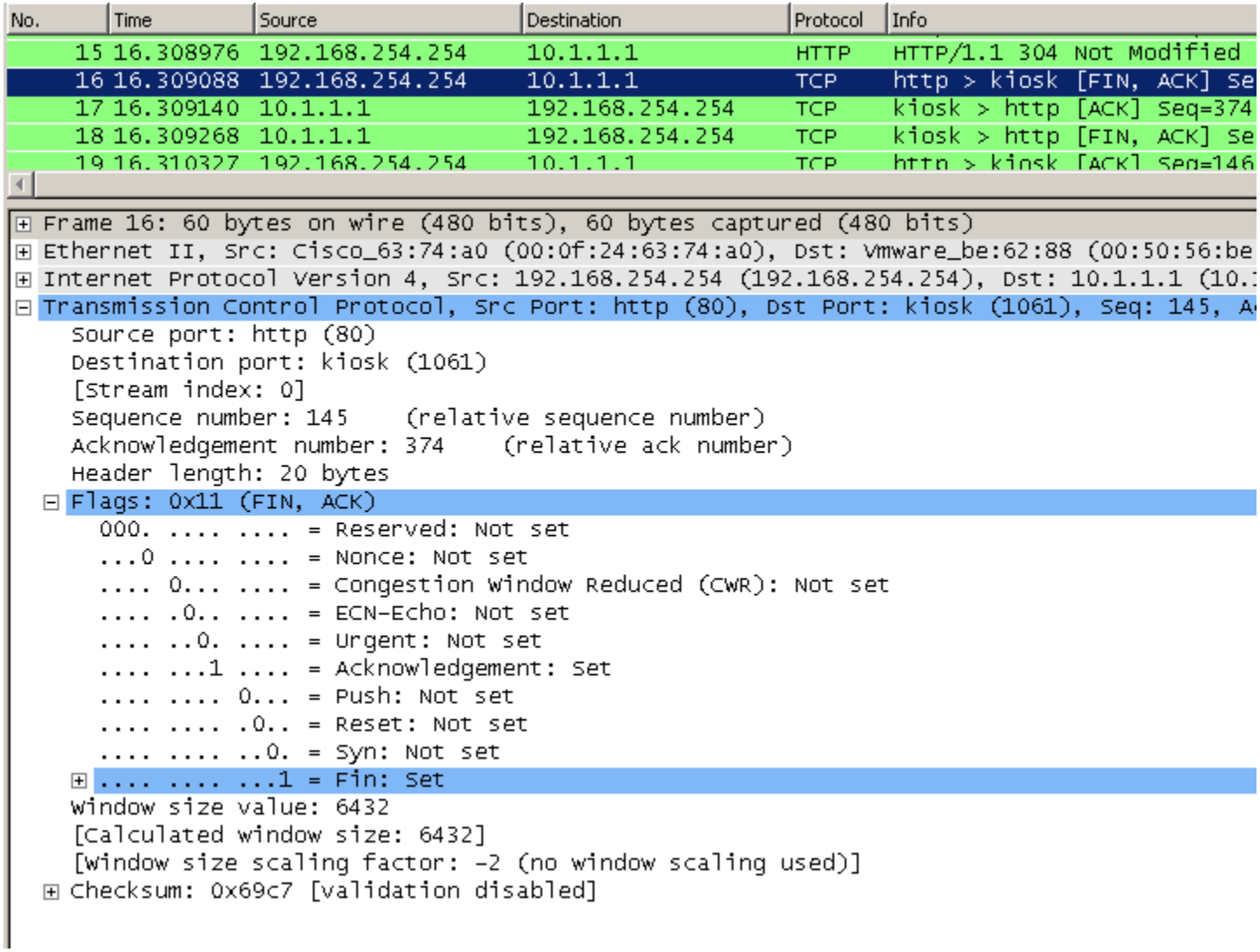
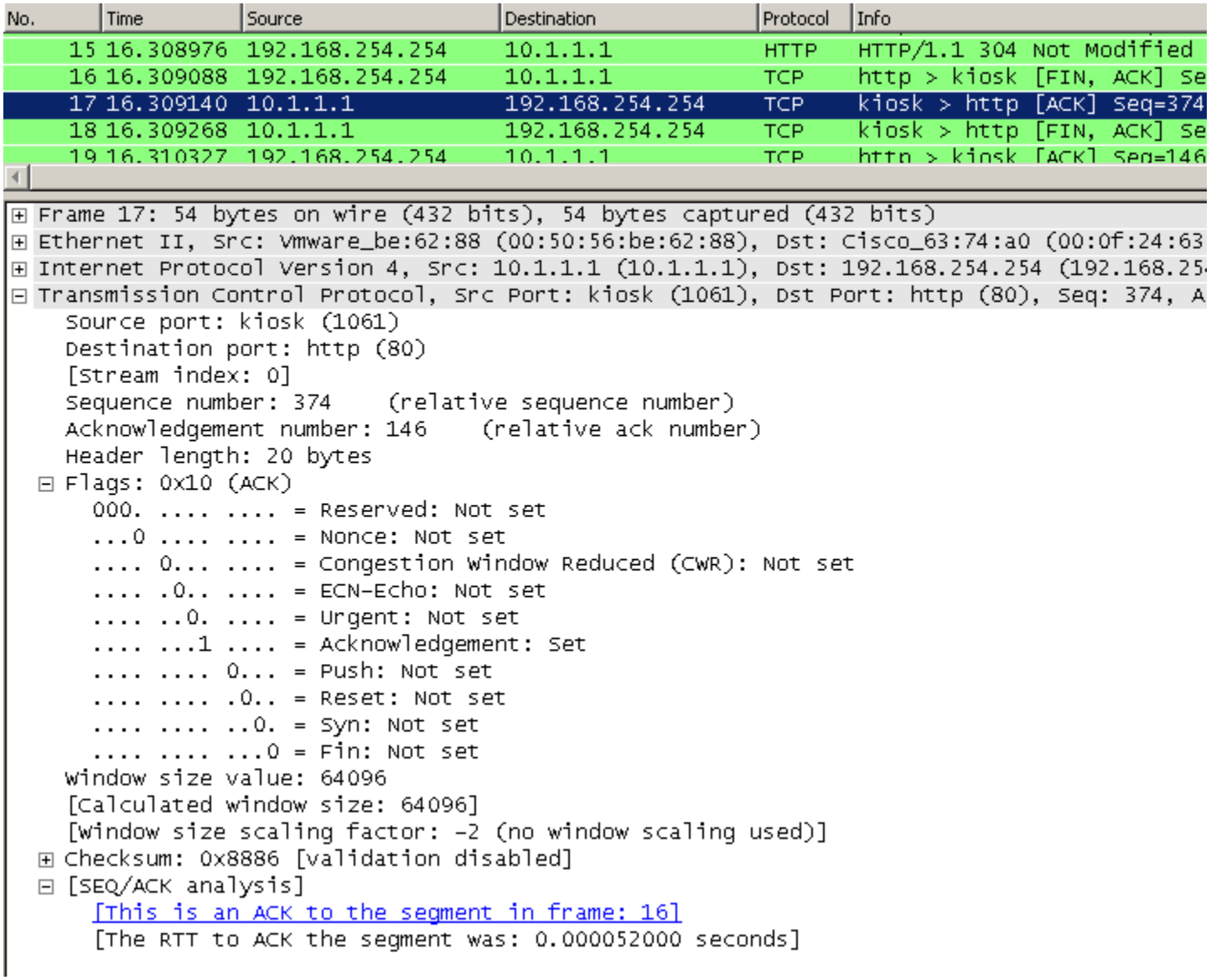


Figure 5 - TCP Session Termination (ACK)



No.	Time	Source	Destination	Protocol	Info
15	16.308976	192.168.254.254	10.1.1.1	HTTP	HTTP/1.1 304 Not Modified
16	16.309088	192.168.254.254	10.1.1.1	TCP	http > kiosk [FIN, ACK] Se
17	16.309140	10.1.1.1	192.168.254.254	TCP	kiosk > http [ACK] Seq=374
18	16.309268	10.1.1.1	192.168.254.254	TCP	kiosk > http [FIN, ACK] Se
19	16.310327	192.168.254.254	10.1.1.1	TCP	http > kiosk [ACK] Seq=146

```
⊞ Frame 17: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
⊞ Ethernet II, Src: Vmware_be:62:88 (00:50:56:be:62:88), Dst: Cisco_63:74:a0 (00:0f:24:63
⊞ Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1), Dst: 192.168.254.254 (192.168.25
⊟ Transmission Control Protocol, Src Port: kiosk (1061), Dst Port: http (80), Seq: 374, A
    Source port: kiosk (1061)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 374    (relative sequence number)
    Acknowledgement number: 146    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x10 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgement: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
    Window size value: 64096
    [Calculated window size: 64096]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0x8886 [validation disabled]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 16]
      [The RTT to ACK the segment was: 0.000052000 seconds]
```