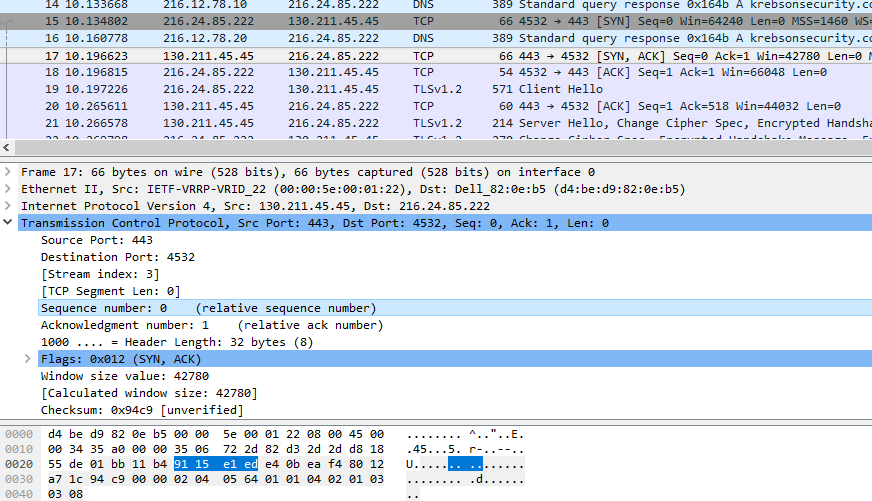
# Lab 6 TCP Handshake, Sequence and Acknowledgement Numbers

TCP uses sequence and acknowledgement numbers to keep track of which bytes and packets have been sent, received, or lost, and to put packets received out of order into the original order.

Sequence and acknowledgement numbers are 4 bytes (32 bits) long, run from 0 - 65535, and start at a random number to make spoofing attacks more difficult. Wireshark displays them in the Packet Details panel as “relative” numbers that always start at 0. The actual number is in the Packet Bytes panel. In the screenshot below, the relative sequence number for this SYN packet is 0, but the actual number is 0x9115e1ed (decimal

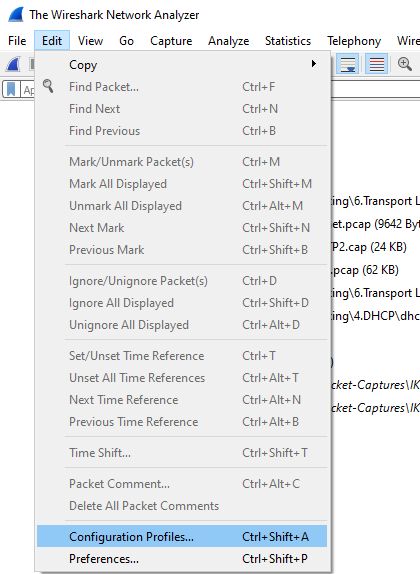
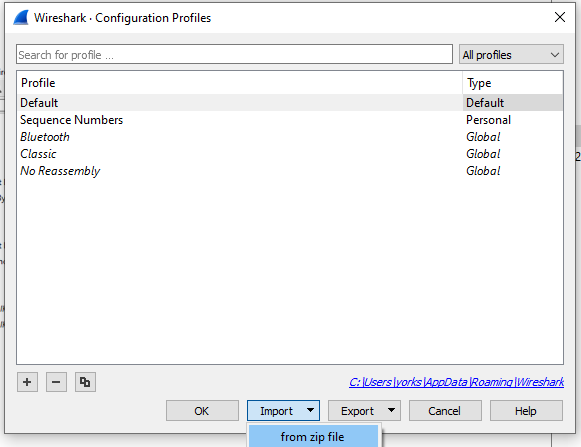


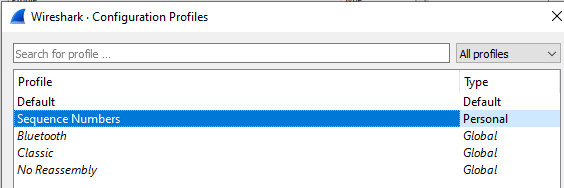
For the rest of this lab we will use the relative numbers from Wireshark, keeping in mind that the actual numbers are different.

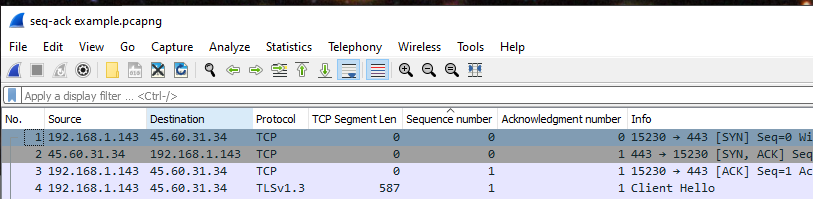
TCP sequence numbers increase by the number of bytes in the TCP data section of the packet; in a way, you can think of them as serial numbers for the bytes sent. SYN and FIN packets are exceptions; they contain no data, but increase the sequence numbers by one. Note that acknowledgement numbers are one higher than the sequence number of the packet the host just received. The host is saying, “The next sequence number I expect to receive is xxx.”

## Modify Wireshark to see Sequence and Acknowledgment Numbers in Action.

For this part of the lab we will use a special configuration to help us understand sequence (seq) and acknowledgement (ack) number. Download the file seq-ack-profile.zip from Canvas. In Wireshark select Configuration Profiles, then Import > from Zip file.

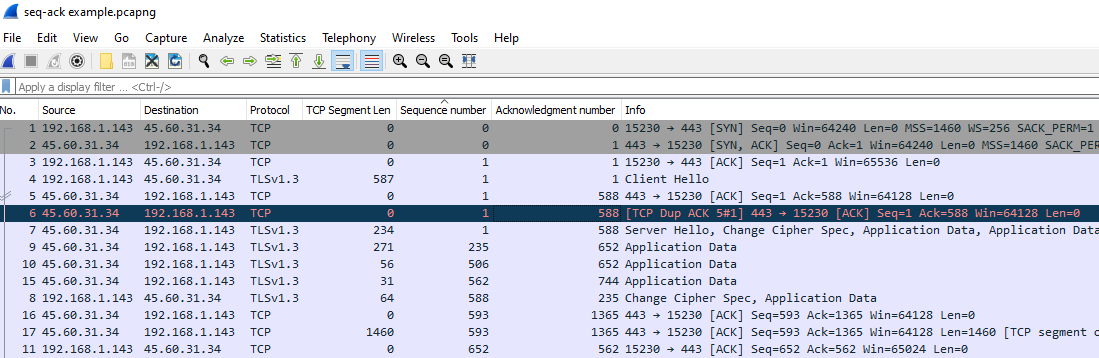
 

Find the seq-ack-profile.zip file and let Wireshark open it. In the same panel (Edit > Configuration Profiles) select the Sequence Numbers profile. Note: To get your Wireshark back to normal after this lab is done, you will need to change the profile back to Default.  


You should see Wireshark columns that look like this.  


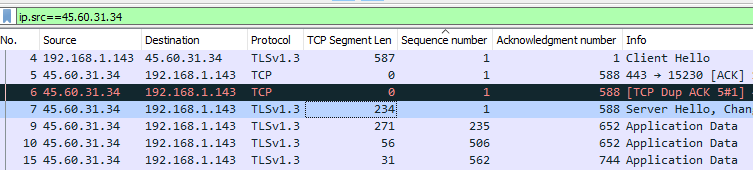
If the profile does not import properly, you can create it using the procedure in the appendix to this lab.

## Packet Capture File

There should be a packet capture file associated with this lab called seq-ack-example.pcapng. Open that file with Wireshark. It is a single stream between 192.168.1.143 and 45.60.31.34, where most of the stream has 45.60.31.34 sending data to 192.168.1.143.  


Note: When Wireshark detects an error in the SEQ or ACK numbers, it highlights the packet in black. Packet 6, above, is a repeat of an ACK packet that 45.60.31.34 in packet 5. Often a duplicate ACK means that the host has missed a packet and wants the packet re-sent. That did not happen in this case, so I am not sure why we received the duplicate packet 6.

### Packets sent by 45.60.31.34



Put ip.src==45.60.31.34 in the display filter so that we only see packets sent by 45.60.31.34 and look at packet 7. The Sequence Number is 1 and the length of the data in the TCP portion (TCP Segment Length) is 234. Then look at the next packet sent by 45.60.31.34, packet 7. The Sequence Number is 235 (234 plus 1), and the TCP Segment Length is 271. In packet 8 the Sequence Number is 506, 235 plus 271.  
A screenshot of a cell phone

Description automatically generated

You can always compute the sequence number of the next packet by adding the sequence number and length of the current packet. The sequence number is basically a

# Turn-in

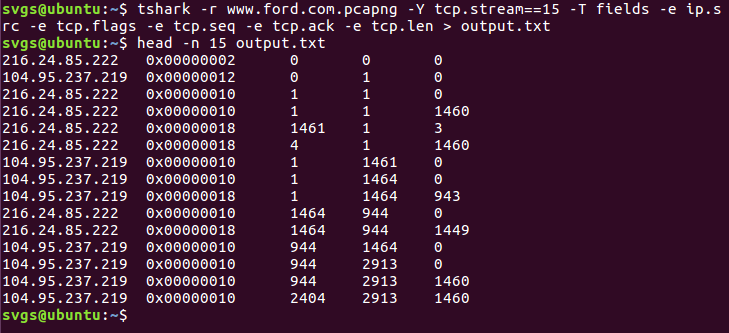
Turn in your worksheet.

## Creating a worksheet automatically (optional)

Wireshark comes with a command line companion called tshark. Wireshark processes large pcap files very slowly; tshark is much faster, although still slow by some standards. We can use tshark and our Linux text editing tools to make a quick, accurate list of sequence and acknowledgement numbers. This example uses a connection to ford.com saved in the file www.ford.com.pcapng. Wireshark was used to examine the file and find that the connection was stream 15 in the file (Follow > TCP Stream).

The command line was:  
tshark -r [www.ford.com.pcap](http://www.ford.com.pcap)ng -Y tcp.stream==15 -T fields -e ip.src -e tcp.flags -e tcp.seq -e tcp.ack -e tcp.len > output.txt

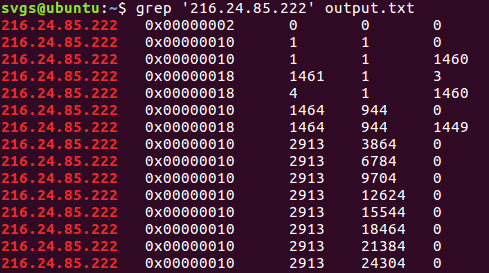
The -r option tells tshark to read from the pcap file. The -Y option, tcp.stream==15, does the same thing as using tcp.stream==15 in the Display Filter window in Wireshark (select stream 15). The -T option tells tshark we only want to extract certain fields, which are listed with the -e options. The final output is redirected to the file output.txt.

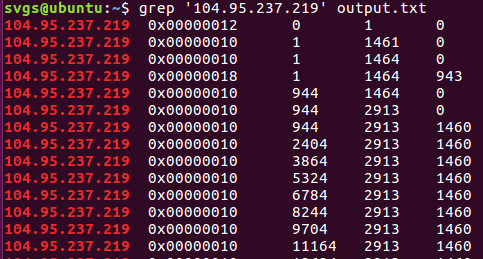
The file output.txt looks like this:  


Headers are:  
ip.src tcp.flags tcp.seq tcp.ack tcp.len

The tcp.flags column can be easily translated. 0x00000002 is SYN, 0x00000012 is SYN/ACK, 0x00000010 is ACK, and 0x00000018 is PSH/ACK. So, 216.24.85.222 starts the handshake with a SYN, 104.95.237.219 replies with a SYN\ACK, and 216.24.85.222 completes the handshake with an ACK.

It is easier to see how the sequence numbers and the packet payload length in bytes are related by looking at the traffic from the two hosts separately. We can do that easily enough with grep.

When we look at traffic from the client 216.24.85.222, we see that once the initial GET request is sent, the sequence number stays the same at 2913 as the client is sending no data to the server, just acknowledgements. The acknowledgement number increments as the client receives traffic from the server. (Note: there is an error in the fourth and fifth packets the client sent. Wireshark interpreted the fifth packet as a TCP Retransmission. The remaining packets are correct.)  
  
Headers are:  
ip.src tcp.flags tcp.seq tcp.ack tcp.len

When we look at the traffic from the server 104.95.237.219, we see that the sequence number begins to increment by 1460 (standard TCP payload for Ethernet is 1460 bytes per packet) for each packet once the file transfer begins. The acknowledgement number stays constant at 2913 once the server has received the GET request from the client.  
  
Headers are:  
ip.src tcp.flags tcp.seq tcp.ack tcp.len

Note: If you’ve gone this far, and read the entire document before you started working, you can submit screenshots for your data instead of filling out a worksheet.

We will use tshark again in later labs that better demonstrate tshark’s power.