Lab Exercise 3

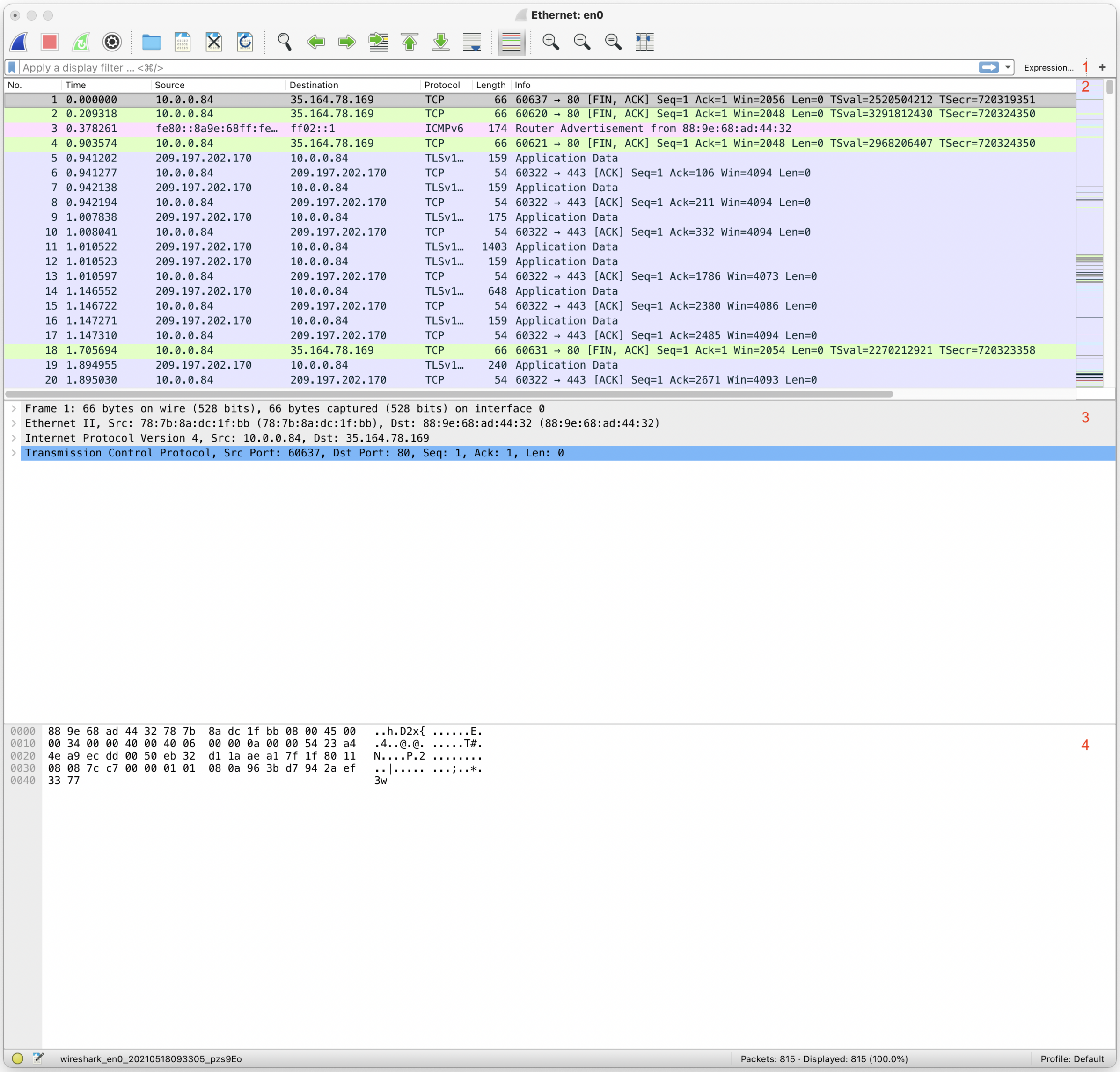
Due: May 26, 2021 **Points:** 100

This laboratory exercise has you use a packet sniffing tool called *wireshark* to see the differences between a web session that uses TLS and one that does not. To do this, you need to install *wireshark*, which can be downloaded from `https://www.wireshark.org`.

Starting

When you start *wireshark*, it brings up a list of interfaces. Choose the one that is appropriate for your system; how you do this depends on what your system is and which interface it uses for the network. For example, on a Mac connected to the network by an ethernet cable, the interface is probably en0. For wireless, look for the Wi-Fi interface.

When *wireshark* starts, the window looks like what is in Figure 1:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.84	35.164.78.169	TCP	66	60637 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2056 Len=0 TSval=2520504212 TSecr=720319351
2	0.209318	10.0.0.84	35.164.78.169	TCP	66	60620 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2048 Len=0 TSval=3291812430 TSecr=720324350
3	0.378261	fe80::8a9e:68ff:fe…	ff02::1	ICMPv6	174	Router Advertisement from 88:9e:68:ad:44:32
4	0.903574	10.0.0.84	35.164.78.169	TCP	66	60621 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2048 Len=0 TSval=2968206407 TSecr=720324350
5	0.941202	209.197.202.170	10.0.0.84	TLSv1…	159	Application Data
6	0.941277	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=106 Win=4094 Len=0
7	0.942138	209.197.202.170	10.0.0.84	TLSv1…	159	Application Data
8	0.942194	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=211 Win=4094 Len=0
9	1.007838	209.197.202.170	10.0.0.84	TLSv1…	175	Application Data
10	1.008041	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=332 Win=4094 Len=0
11	1.010522	209.197.202.170	10.0.0.84	TLSv1…	1403	Application Data
12	1.010523	209.197.202.170	10.0.0.84	TLSv1…	159	Application Data
13	1.010597	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=1786 Win=4073 Len=0
14	1.146552	209.197.202.170	10.0.0.84	TLSv1…	648	Application Data
15	1.146722	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=2380 Win=4086 Len=0
16	1.147271	209.197.202.170	10.0.0.84	TLSv1…	159	Application Data
17	1.147310	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=2485 Win=4094 Len=0
18	1.705694	10.0.0.84	35.164.78.169	TCP	66	60631 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2054 Len=0 TSval=2270212921 TSecr=720323358
19	1.894955	209.197.202.170	10.0.0.84	TLSv1…	240	Application Data
20	1.895030	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=2671 Win=4093 Len=0

```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: 78:7b:8a:dc:1f:bb (78:7b:8a:dc:1f:bb), Dst: 88:9e:68:ad:44:32 (88:9e:68:ad:44:32)
> Internet Protocol Version 4, Src: 10.0.0.84, Dst: 35.164.78.169
> Transmission Control Protocol, Src Port: 60637, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
```

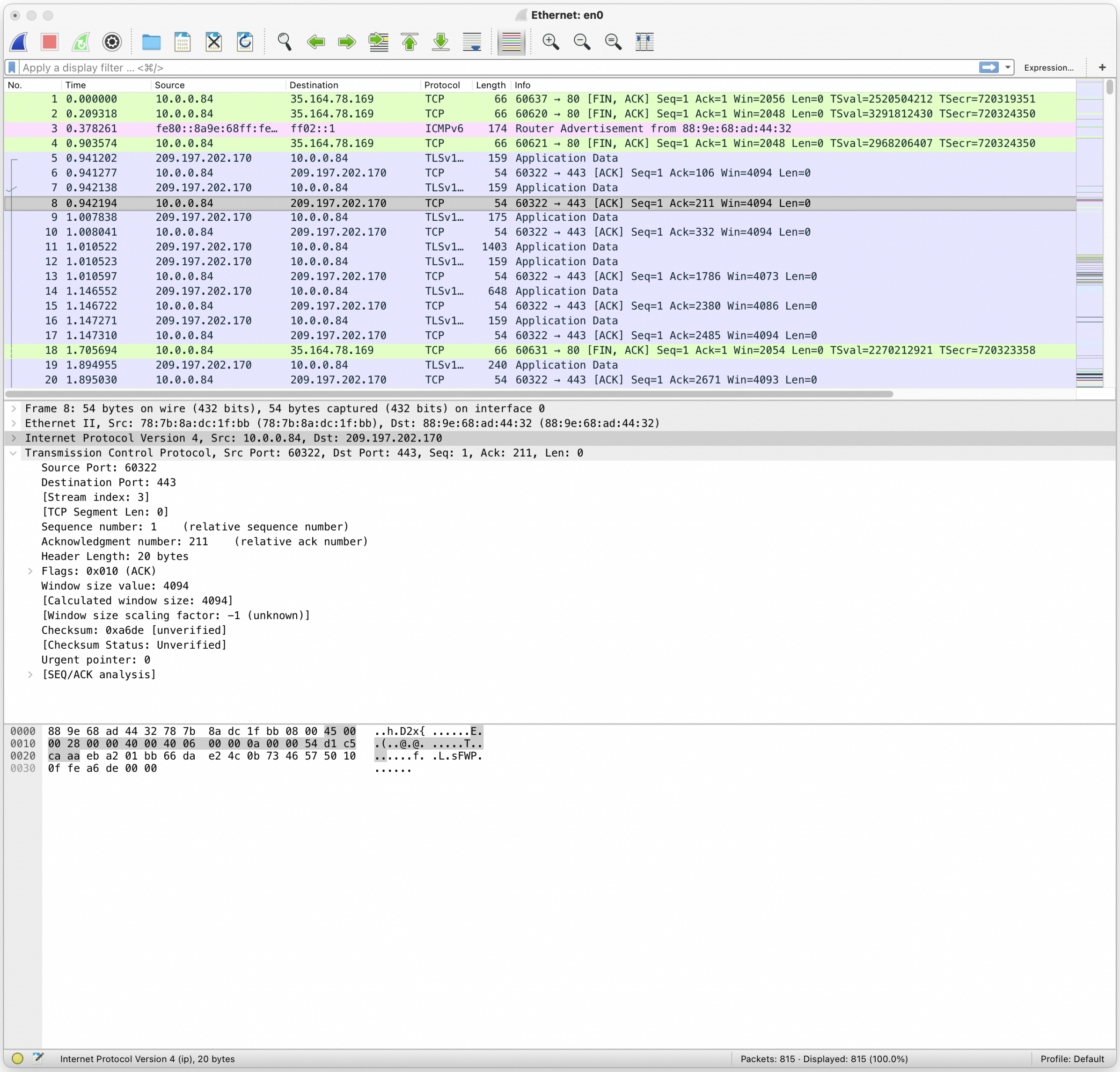
```
0000  88 9e 68 ad 44 32 78 7b  8a dc 1f bb 08 00 45 00   ..h.D2x{ ......E.
0010  00 34 00 00 40 00 40 06  00 00 0a 00 00 54 23 a4   .4..@.@. .....T#.
0020  4e a9 ec dd 00 50 eb 32  d1 1a ae a1 7f 1f 80 11   N....P.2 ........
0030  08 08 7c c7 00 00 01 01  08 0a 96 3b d7 94 2a ef   ..|..... ...;..*.
0040  33 77                                              3w
```

wireshark_en0_20210518093305_pzs9Eo Packets: 815 · Displayed: 815 (100.0%) Profile: Default

Figure 1: Initial widow from *wireshark*

For our purposes, the important parts are the following:

1. The filter bar, just under the top bar with the icons;
2. Some columns that we will use are in the window under the filter bar. The ones we're interested in are:
   - No., which is the packet number assigned by *wireshark*;
   - Source, which is the origin of the packet; for TCP/IP, this is an IP address;
   - Destination, which is where the packet is going; again, for TCP/IP, this is an IP address;
   - Protocol, which is the TCP or IP protocol; and
   - Info, which contains information about the packet.
3. The next window contains information about the various layers (envelopes); note the > signs to the left of each layer. If you click on them, they will expand the information at that layer.
4. The bottom window is the contents of the packet. The grey area has byte numbers in hex; the next two columns, each with 8 byte representations, have the contents of the packet in hex. To the right of those is the same information but as a printable character or a "." (meaning either the byte is non-printing or a period).



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.84	35.164.78.169	TCP	66	60637 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2056 Len=0 TSval=2520504212 TSecr=720319351
2	0.209318	10.0.0.84	35.164.78.169	TCP	66	60620 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2048 Len=0 TSval=3291812430 TSecr=720324350
3	0.378261	fe80::8a9e:68ff:fe…	ff02::1	ICMPv6	174	Router Advertisement from 88:9e:68:ad:44:32
4	0.903574	10.0.0.84	35.164.78.169	TCP	66	60621 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2048 Len=0 TSval=2968206407 TSecr=720324350
5	0.941202	209.197.202.170	10.0.0.84	TLSv1…	159	Application Data
6	0.941277	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=106 Win=4094 Len=0
7	0.942138	209.197.202.170	10.0.0.84	TLSv1…	159	Application Data
8	0.942194	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=211 Win=4094 Len=0
9	1.007838	209.197.202.170	10.0.0.84	TLSv1…	175	Application Data
10	1.008041	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=332 Win=4094 Len=0
11	1.010522	209.197.202.170	10.0.0.84	TLSv1…	1403	Application Data
12	1.010523	209.197.202.170	10.0.0.84	TLSv1…	159	Application Data
13	1.010597	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=1786 Win=4073 Len=0
14	1.146552	209.197.202.170	10.0.0.84	TLSv1…	648	Application Data
15	1.146722	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=2380 Win=4086 Len=0
16	1.147271	209.197.202.170	10.0.0.84	TLSv1…	159	Application Data
17	1.147310	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=2485 Win=4094 Len=0
18	1.705694	10.0.0.84	35.164.78.169	TCP	66	60631 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2054 Len=0 TSval=2270212921 TSecr=720323358
19	1.894955	209.197.202.170	10.0.0.84	TLSv1…	240	Application Data
20	1.895030	10.0.0.84	209.197.202.170	TCP	54	60322 → 443 [ACK] Seq=1 Ack=2671 Win=4093 Len=0

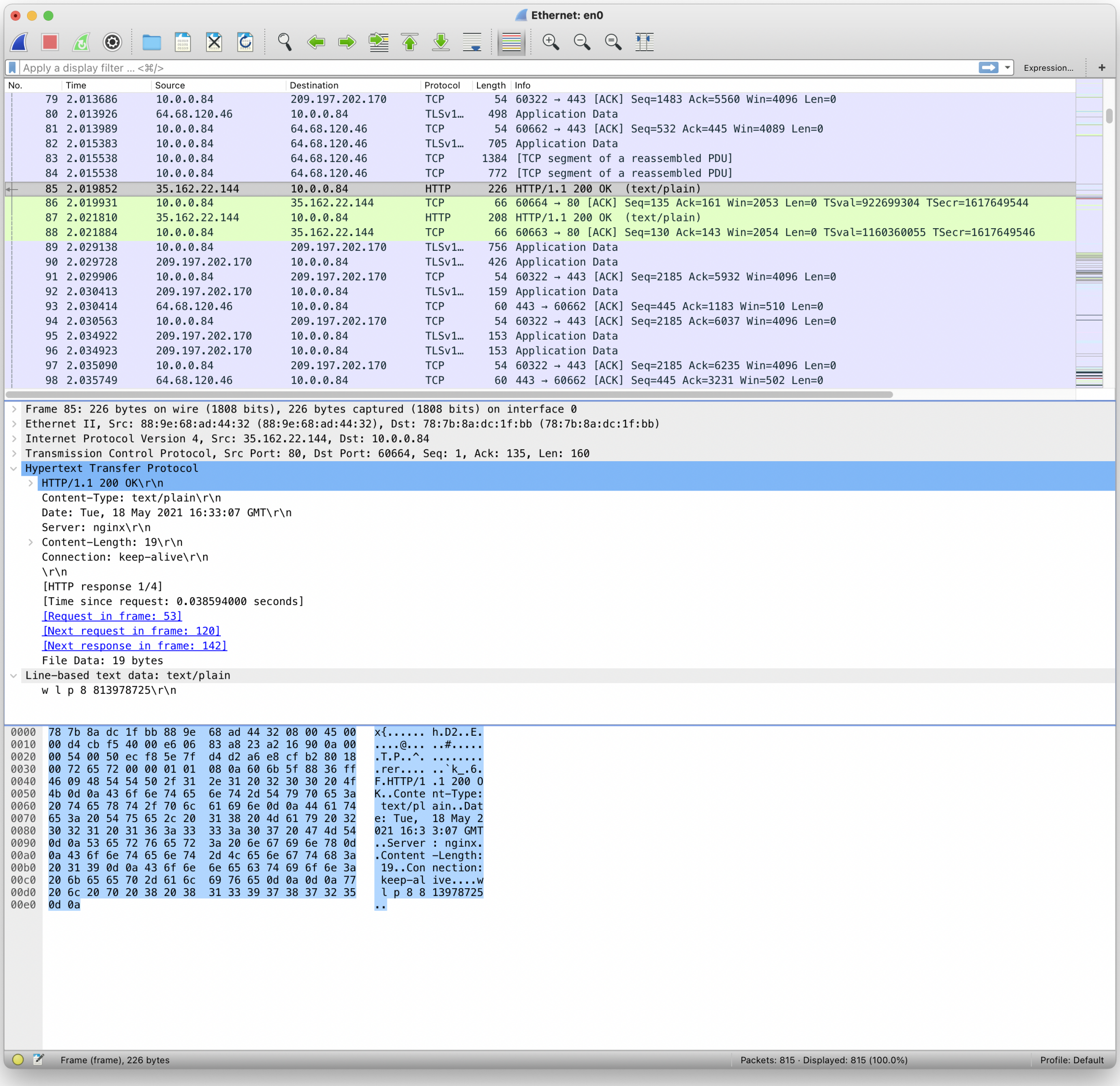
```
> Frame 8: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: 78:7b:8a:dc:1f:bb (78:7b:8a:dc:1f:bb), Dst: 88:9e:68:ad:44:32 (88:9e:68:ad:44:32)
> Internet Protocol Version 4, Src: 10.0.0.84, Dst: 209.197.202.170
v Transmission Control Protocol, Src Port: 60322, Dst Port: 443, Seq: 1, Ack: 211, Len: 0
    Source Port: 60322
    Destination Port: 443
    [Stream index: 3]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 211    (relative ack number)
    Header Length: 20 bytes
  > Flags: 0x010 (ACK)
    Window size value: 4094
    [Calculated window size: 4094]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xa6de [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
```

```
0000  88 9e 68 ad 44 32 78 7b  8a dc 1f bb 08 00 45 00   ..h.D2x{ ......E.
0010  00 28 00 00 40 00 40 06  00 00 0a 00 00 54 d1 c5   .(..@.@. .....T..
0020  ca aa eb a2 01 bb 66 da  e2 4c 0b 73 46 57 50 10   ......f. .L.sFWP.
0030  0f fe a6 de 00 00                                  ......
```

Internet Protocol Version 4 (ip), 20 bytes | Packets: 815 · Displayed: 815 (100.0%) | Profile: Default

Figure 2: Expansion of TCP part of packet 8

Now look at Figure 2. This is the same window but with packet 8's TCP layer expanded. The origin and destination can be read either from the *wireshark* list of packets, or from the Internet Protocol line (which can itself be expanded by clicking on the > to the left of it).


Ethernet: en0
Apply a display filter ... <⌘/>
Expression...
+
Frame (frame), 226 bytes
Packets: 815 · Displayed: 815 (100.0%)
Profile: Default

```
No.  Time      Source           Destination      Protocol Length Info
  79 2.013686  10.0.0.84        209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=1483 Ack=5560 Win=4096 Len=0
  80 2.013926  64.68.120.46     10.0.0.84        TLSv1…    498 Application Data
  81 2.013989  10.0.0.84        64.68.120.46     TCP        54 60662 → 443 [ACK] Seq=532 Ack=445 Win=4089 Len=0
  82 2.015383  10.0.0.84        64.68.120.46     TLSv1…    705 Application Data
  83 2.015538  10.0.0.84        64.68.120.46     TCP      1384 [TCP segment of a reassembled PDU]
  84 2.015538  10.0.0.84        64.68.120.46     TCP       772 [TCP segment of a reassembled PDU]
  85 2.019852  35.162.22.144    10.0.0.84        HTTP      226 HTTP/1.1 200 OK  (text/plain)
  86 2.019931  10.0.0.84        35.162.22.144    TCP        66 60664 → 80 [ACK] Seq=135 Ack=161 Win=2053 Len=0 TSval=922699304 TSecr=1617649544
  87 2.021810  35.162.22.144    10.0.0.84        HTTP      208 HTTP/1.1 200 OK  (text/plain)
  88 2.021884  10.0.0.84        35.162.22.144    TCP        66 60663 → 80 [ACK] Seq=130 Ack=143 Win=2054 Len=0 TSval=1160360055 TSecr=1617649546
  89 2.029138  10.0.0.84        209.197.202.170  TLSv1…    756 Application Data
  90 2.029728  209.197.202.170  10.0.0.84        TLSv1…    426 Application Data
  91 2.029906  10.0.0.84        209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=2185 Ack=5932 Win=4096 Len=0
  92 2.030413  209.197.202.170  10.0.0.84        TLSv1…    159 Application Data
  93 2.030414  64.68.120.46     10.0.0.84        TCP        60 443 → 60662 [ACK] Seq=445 Ack=1183 Win=510 Len=0
  94 2.030563  10.0.0.84        209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=2185 Ack=6037 Win=4096 Len=0
  95 2.034922  209.197.202.170  10.0.0.84        TLSv1…    153 Application Data
  96 2.034923  209.197.202.170  10.0.0.84        TLSv1…    153 Application Data
  97 2.035090  10.0.0.84        209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=2185 Ack=6235 Win=4096 Len=0
  98 2.035749  64.68.120.46     10.0.0.84        TCP        60 443 → 60662 [ACK] Seq=445 Ack=3231 Win=502 Len=0
```

```
> Frame 85: 226 bytes on wire (1808 bits), 226 bytes captured (1808 bits) on interface 0
> Ethernet II, Src: 88:9e:68:ad:44:32 (88:9e:68:ad:44:32), Dst: 78:7b:8a:dc:1f:bb (78:7b:8a:dc:1f:bb)
> Internet Protocol Version 4, Src: 35.162.22.144, Dst: 10.0.0.84
> Transmission Control Protocol, Src Port: 80, Dst Port: 60664, Seq: 1, Ack: 135, Len: 160
v Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Content-Type: text/plain\r\n
    Date: Tue, 18 May 2021 16:33:07 GMT\r\n
    Server: nginx\r\n
  > Content-Length: 19\r\n
    Connection: keep-alive\r\n
    \r\n
    [HTTP response 1/4]
    [Time since request: 0.038594000 seconds]
    [Request in frame: 53]
    [Next request in frame: 120]
    [Next response in frame: 142]
    File Data: 19 bytes
v Line-based text data: text/plain
    w l p 8 813978725\r\n
```

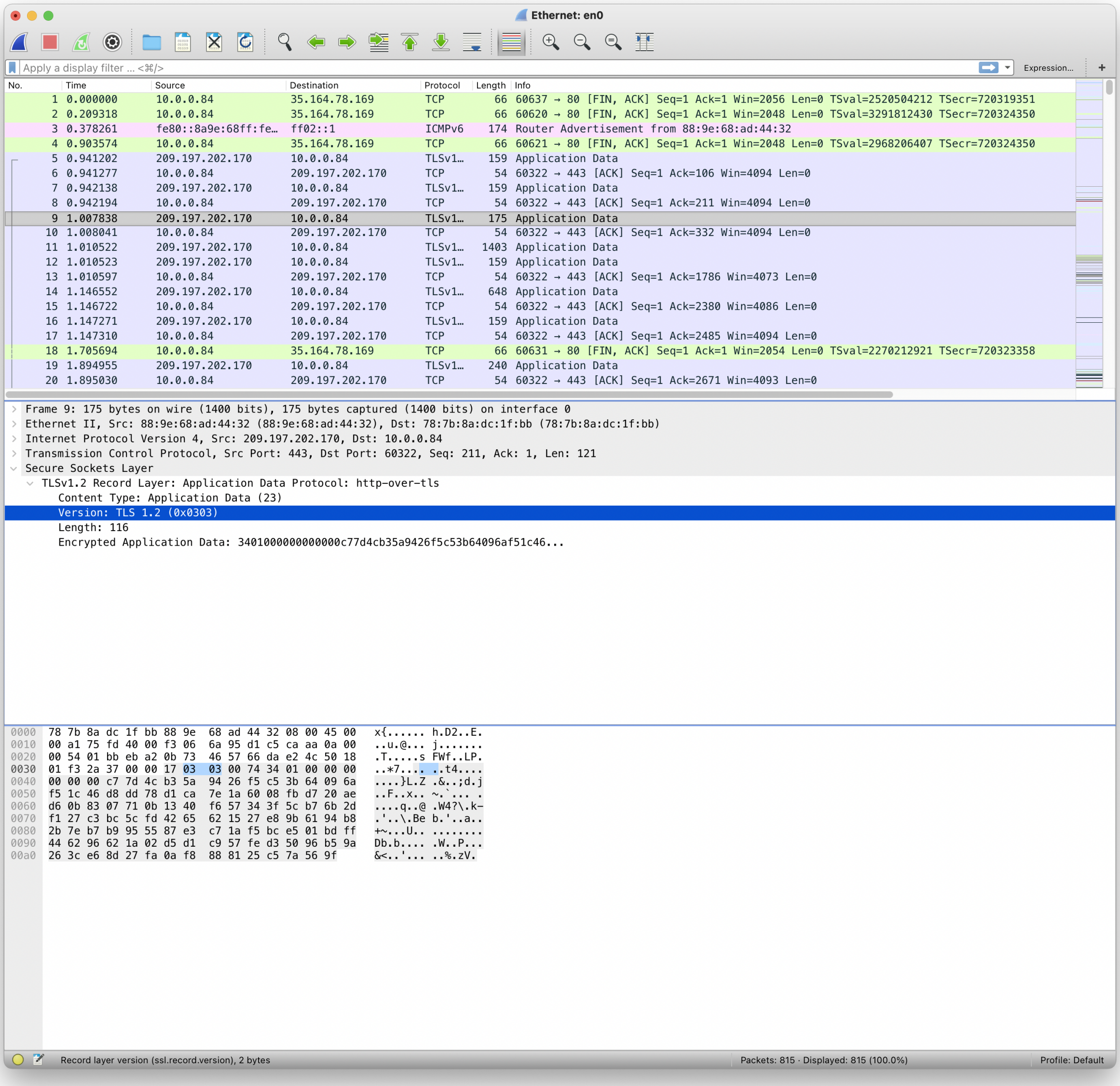
```
0000  78 7b 8a dc 1f bb 88 9e  68 ad 44 32 08 00 45 00   x{...... h.D2..E.
0010  00 d4 cb f5 40 00 e6 06  83 a8 23 a2 16 90 0a 00   ....@... ..#.....
0020  00 54 00 50 ec f8 5e 7f  d4 d2 a6 e8 cf b2 80 18   .T.P..^. ........
0030  00 72 65 72 00 00 01 01  08 0a 60 6b 5f 88 36 ff   .rer.... ..`k_.6.
0040  46 09 48 54 54 50 2f 31  2e 31 20 32 30 30 20 4f   F.HTTP/1 .1 200 O
0050  4b 0d 0a 43 6f 6e 74 65  6e 74 2d 54 79 70 65 3a   K..Conte nt-Type:
0060  20 74 65 78 74 2f 70 6c  61 69 6e 0d 0a 44 61 74    text/pl ain..Dat
0070  65 3a 20 54 75 65 2c 20  31 38 20 4d 61 79 20 32   e: Tue,  18 May 2
0080  30 32 31 20 31 36 3a 33  33 3a 30 37 20 47 4d 54   021 16:3 3:07 GMT
0090  0d 0a 53 65 72 76 65 72  3a 20 6e 67 69 6e 78 0d   ..Server : nginx.
00a0  0a 43 6f 6e 74 65 6e 74  2d 4c 65 6e 67 74 68 3a   .Content -Length:
00b0  20 31 39 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a    19..Con nection:
00c0  20 6b 65 65 70 2d 61 6c  69 76 65 0d 0a 0d 0a 77    keep-al ive....w
00d0  20 6c 20 70 20 38 20 38  31 33 39 37 38 37 32 35    l p 8 8 13978725
00e0  0d 0a                                              ..
```

Figure 3: Expansion of HTTP part of packet 85

Figure 3 shows an application layer protocol, HTTP, on top of the TCP protocol. In this figure, it is expanded. Notice the name of the server there, "nginx" followed by a carriage return (\r) and line feed (\n). Now look in the window below, at the character representation of the packet. Byte 0x92 begins the same line, "Server nginx" (the carriage return and newline are non-printing, so the two "."s following the string are them. The important point is you can read the text in the packet directly. Indeed, *wireshark* simply does so and extracts information to construct the representation of the Hypertext Transfer Protocol.

Figure 4 shows the expansion of packet 9. It is similar to that of packet 8 (see Figure 2), except that below the Transmission Control Protocol line is a Secure Sockets Layer line. If you click on the > for that, another line, the TLSv1.2 Record Layer, will appear. Click on the > for that, and you see information about the packet.

If you want to see the bytes that correspond to the fields, click on the field. The corresponding bytes (and characters) in the packet will be highlighted in the packet body. Figure 4 shows this. The version is selected, and you can see the dark highlighting in the packet body. The light highlighting corresponds to the secure socket layer part of the packet.



```
No.  Time      Source              Destination      Protocol Length Info
   1 0.000000  10.0.0.84           35.164.78.169    TCP        66 60637 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2056 Len=0 TSval=2520504212 TSecr=720319351
   2 0.209318  10.0.0.84           35.164.78.169    TCP        66 60620 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2048 Len=0 TSval=3291812430 TSecr=720324350
   3 0.378261  fe80::8a9e:68ff:fe… ff02::1          ICMPv6    174 Router Advertisement from 88:9e:68:ad:44:32
   4 0.903574  10.0.0.84           35.164.78.169    TCP        66 60621 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2048 Len=0 TSval=2968206407 TSecr=720324350
   5 0.941202  209.197.202.170     10.0.0.84        TLSv1…    159 Application Data
   6 0.941277  10.0.0.84           209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=1 Ack=106 Win=4094 Len=0
   7 0.942138  209.197.202.170     10.0.0.84        TLSv1…    159 Application Data
   8 0.942194  10.0.0.84           209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=1 Ack=211 Win=4094 Len=0
   9 1.007838  209.197.202.170     10.0.0.84        TLSv1…    175 Application Data
  10 1.008041  10.0.0.84           209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=1 Ack=332 Win=4094 Len=0
  11 1.010522  209.197.202.170     10.0.0.84        TLSv1…   1403 Application Data
  12 1.010523  209.197.202.170     10.0.0.84        TLSv1…    159 Application Data
  13 1.010597  10.0.0.84           209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=1 Ack=1786 Win=4073 Len=0
  14 1.146552  209.197.202.170     10.0.0.84        TLSv1…    648 Application Data
  15 1.146722  10.0.0.84           209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=1 Ack=2380 Win=4086 Len=0
  16 1.147271  209.197.202.170     10.0.0.84        TLSv1…    159 Application Data
  17 1.147310  10.0.0.84           209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=1 Ack=2485 Win=4094 Len=0
  18 1.705694  10.0.0.84           35.164.78.169    TCP        66 60631 → 80 [FIN, ACK] Seq=1 Ack=1 Win=2054 Len=0 TSval=2270212921 TSecr=720323358
  19 1.894955  209.197.202.170     10.0.0.84        TLSv1…    240 Application Data
  20 1.895030  10.0.0.84           209.197.202.170  TCP        54 60322 → 443 [ACK] Seq=1 Ack=2671 Win=4093 Len=0
```

```
> Frame 9: 175 bytes on wire (1400 bits), 175 bytes captured (1400 bits) on interface 0
> Ethernet II, Src: 88:9e:68:ad:44:32 (88:9e:68:ad:44:32), Dst: 78:7b:8a:dc:1f:bb (78:7b:8a:dc:1f:bb)
> Internet Protocol Version 4, Src: 209.197.202.170, Dst: 10.0.0.84
> Transmission Control Protocol, Src Port: 443, Dst Port: 60322, Seq: 211, Ack: 1, Len: 121
v Secure Sockets Layer
  v TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 116
      Encrypted Application Data: 3401000000000000c77d4cb35a9426f5c53b64096af51c46...
```

```
0000  78 7b 8a dc 1f bb 88 9e  68 ad 44 32 08 00 45 00   x{...... h.D2..E.
0010  00 a1 75 fd 40 00 f3 06  6a 95 d1 c5 ca aa 0a 00   ..u.@... j.......
0020  00 54 01 bb eb a2 0b 73  46 57 66 da e2 4c 50 18   .T.....s FWf..LP.
0030  01 f3 2a 37 00 00 17 03  03 00 74 34 01 00 00 00   ..*7.... ..t4....
0040  00 00 00 c7 7d 4c b3 5a  94 26 f5 c5 3b 64 09 6a   ....}L.Z .&..;d.j
0050  f5 1c 46 d8 dd 78 d1 ca  7e 1a 60 08 fb d7 20 ae   ..F..x.. ~.`... .
0060  d6 0b 83 07 71 0b 13 40  f6 57 34 3f 5c b7 6b 2d   ....q..@ .W4?\.k-
0070  f1 27 c3 bc 5c fd 42 65  62 15 27 e8 9b 61 94 b8   .'..\.Be b.'..a..
0080  2b 7e b7 b9 95 55 87 e3  c7 1a f5 bc e5 01 bd ff   +~...U.. ........
0090  44 62 96 62 1a 02 d5 d1  c9 57 fe d3 50 96 b5 9a   Db.b.... .W..P...
00a0  26 3c e6 8d 27 fa 0a f8  88 81 25 c5 7a 56 9f      &<..'... ..%.zV.
```

Record layer version (ssl.record.version), 2 bytes   Packets: 815 · Displayed: 815 (100.0%)   Profile: Default

Figure 4: Expansion of TLS part of packet 9

The encrypted data begins at position 0x3b in the packet body (look at the lowest window). Notice you can't read any words; this is because it is encrypted. That's the difference between HTTP and HTTPS (which is HTTP over TLS).

Given all this, we're ready to go ahead with the lab.

Lab Exercise 1

First, open *wireshark* and select the network interface. Next, open your web browser (if it is already open, close it) and go to `https://www.cnn.com`. Notice the look next to the URL; this says the connection is secured using TLS. Let's look for the packers related to the connection. To do this, we filter the packets:

ip.addr==*a.b.c.d*

where *a.b.c.d* is the IP address of your system (be sure to hit return or enter after typing the above; the bar should turn green). If you are behind a NAT, use the address of your system, not the one exposed to the outside word. This eliminates any packets not going to, or coming from, your system.

You can click the big red button now to stop *wireshark* from getting packets. You don't have to, but it may make reading things easier.

Now let's look for the connection to www.cnn.com. Add "&&tls" to the filter, again pressing return or enter. The Protocol column should have lines beginning with "TLS" only. Look down the packet list until you see ”Client Hello” in the info field. This is the start of a TLS connection.

Now look for ”Client Hello” in the Info field. Search for it by clicking on the magnifying glass to the upper left; it is probably next to a green arrow). Then between the ”Case Sensitive” checkbox (leave this unchecked) and the search window is a drop-down menu. Chang it to ”String” and type ”Client Hello” in the search box, and click Find.

In the bottom window, look for ”www.cnn.com”. This will be in the rightmost two columns, and may well be split over them (the columns read across). If you see another host name there, click Find again, and repeat until you see the www.cnn.com You may have to click through several packets before finding it (I found it on the 20th click).

Now get the destination address from this packet; call it *w.x.y.z*. That's the address for www.cnn.com. Do *not* use a DNS lookup to get the IP address. You will get several, and there is no guarantee this one will be among them. Add that address to the filter as "&&ip.addr==*w.x.y.z*" and press return. Now you will see all the packets exchanged between your system and www.cnn.com.

Let's see what ciphers your browser recognizes. Go to the middle window and click on the > sign next to Transport Layer Security. Click on the > sign next to the Record Layer line that appears below it, and then on the > sign next to the Handshake Protocol. That contains the information from the handshake protocol such as the ciphers the client can speak.

Client Hello Please answer the following questions, and include a screen shot showing how you got your answers:

1a. What are the source and destination IP addresses?
1b. What is the source port on the client (your system), and the destination port on the peer (the server)?
1c. How many ciphers will the client accept?
1d. How many compression methods will the client accept?

Server Hello Now answer these about the server. The information will be in the Server Hello packet. Again, include a screen shot showing how you got your answers

2a. What are the source and destination IP addresses of the Server Hello packet?
2b. What is the source port on the peer (the server), and the destination port on the client (your system)?
2c. Which cipher does the server select?
2d. Which compression method does the server select?

Certificates Next, go to the Certificate packet. Please answer these questions and include a screen shot showing how you got your answer.

3a. How many certificates are present?
3b. Are these going from the client to the server or the server to the client? How do you know?
3c. Look at the first certificate. What algorithm is being used? What subject was the signed certificate issued to?

Key Exchange Algorithm Now let's see what the key exchange algorithm is. Look at the Server Key Exchange packet. Please answer this question and include a screen shot showing how you got your answer.

4. What cryptographic handshake algorithm is being used? You need not give the parameters.

Cipher Spec Look at the Change Cipher Spec packet. Please answer this question and include a screen shot showing how you got your answer.

5. What is the length of this message (not the packet, the Change Cipher Spec message)?

Application Data Finally, look at the first Application Data packet. Please answer this question and include a screen shot showing how you got your answer.

6a. What is the Content Type of this packet?
6b. Can you read any of the application-level data?

Lab Exercise 2

Now restart *wireshark* so it will gather packets. Close your web browser and reopen it. Go to the host `http://nob.cs.ucdavis.edu`, IP address 169.237.6.105.

7a. What is the filter to display only the packets between your host and nob.cs.ucdavis.edu?

7b. What is the filter to display only the HTTP ones?

One of the packets says “404 Not Found”. Please answer this question and include a screen shot showing how you got your answer.

8. What file was not found?

Locate the packet number (far left column). Change the filter to display TCP packets. Now, look at the packets preceding the “not found” one. Please answer this question and include a screen shot showing how you got your answer.

9. One of the packets contains the name of the web server program (like *apache* or *nginx*) on nob.cs.ucdavis.edu. What is it?