
VANIER COLLEGE – Computer Engineering Technology – Autumn 2021

Network Systems Design (247-509-VA)

Leonardo Fusser (1946995)

LABORATORY EXPERIMENT 4

Application and Transport layer protocols

NOTE:

To be completed in one lab session of 3 hrs.

To be submitted at the end of the lab session.

This exercise is to be done individually except where specified in the procedure. **Each** student must submit the lab results with original observations and conclusions.

Name : Leonardo Fusser

Student Number : 1946995

OBJECTIVES:

After performing this experiment, the student will be able to:

1. Configure the host computer to capture application layer protocols.
2. Capture and analyze HTTP communication between the host computer and a web server.
3. Capture and analyze FTP communication between the host computer and a FTP server.
4. Observe TCP establish and manage communication channels with HTTP and FTP connections.

THEORY

The primary function of the transport layer is to keep track of multiple application conversations on the same host. However, different applications have different requirements for their data, and therefore different transport protocols have been developed to meet these requirements.

Application layer protocols define the communication between network services, such as a web server and client, and an FTP server and client. Clients initiate communication to the appropriate server, and the server responds to the client. For each network service, a different server is listening on a different port for client connections. There may be several servers on the same end device. A user may open several client applications to the same server, yet each client communicates exclusively with a session established between the client and server.

Application layer protocols rely on lower-level TCP/IP protocols, such as TCP and UDP. This lab examines two popular application layer protocols, HTTP and FTP, and how transports layer protocols TCP and UDP manage the communication channel. Also examined are popular client requests and corresponding server responses.

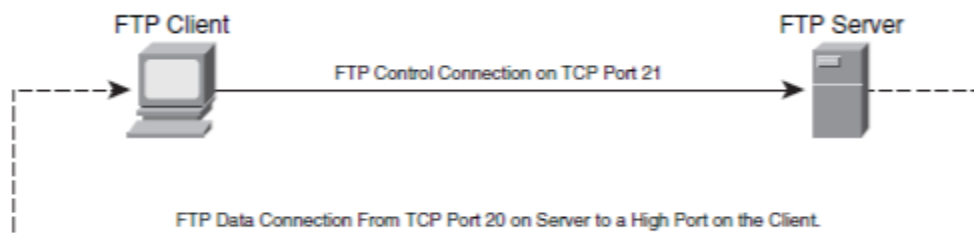
In this lab, you use client applications to connect to Eagle Server network services. You monitor the communication with Wireshark and analyze the captured packets.

The web browser will also be used to examine the FTP protocol, as will the FTP command-line client. This exercise demonstrates that although clients may differ, the underlying communication to the server remains the same.

FTP active and passive transfer modes

The implications between the two modes are very important from an information security perspective. The transfer mode sets how the data port is configured.

In active transfer mode, a client initiates an FTP session with the server on well-known TCP port 21. For data transfer, the server initiates a connection from well-known TCP port 20 to a client's high port, a port number above 1023.



Unless the FTP client firewall is configured to permit connections from the outside, data transfer may fail. To establish connectivity for data transfer, the FTP client must permit either FTP-related connections (implying stateful packet filtering), or disable blocking.

In passive transfer mode, a client initiates an FTP session with the server on well-known TCP port 21 (same connection used in the active transfer mode). For data transfer, however, there are two significant changes. First, the client initiates the data connection to the server. Second, high ports are used on both ends of the connection.



Unless the FTP server is configured to permit a connection to a random high port, data transfer will fail. Not all FTP client applications support changes to the transfer mode.

PROCEDURE

(Modified based on CCNA's labs)

The lab will be using the same Eagle-server Topology Diagram as in our previous labs. If you have changed your computer, please modify your network IP address accordingly.

Part A: Capture and analyze HTTP communication between host and a web server

HTTP is an application layer protocol, relying on lower-level protocols such as TCP to establish and manage the communication channel.

- 1) Start Wireshark captures. Then, start a web browser on your host computer, connect to URL `http://eagle-server.example.com`.
 - a) Click the web browser **Refresh** button. There should be no change in the display.
 - b) Open a second web browser and connect to URL `http://eagle-server.example.com/page2.html`. This will display a different web page.
 - c) Do not close either browser until Wireshark capture is stopped.

2) Stop Wireshark, close the web browsers and analyze the captured data.

- a) There are at least 3 HTTP sessions created in Step 1. Identify and briefly explain them based on your captured data.

No.	Time	Source	Destination	Protocol	Length	Info
424	52.480631	172.16.22.1	192.168.254.254	HTTP	346	GET / HTTP/1.1
426	52.484041	192.168.254.254	172.16.22.1	HTTP	502	HTTP/1.1 200 OK (text/html)
434	52.535889	172.16.22.1	192.168.254.254	HTTP	287	GET /favicon.ico HTTP/1.1
436	52.539037	192.168.254.254	172.16.22.1	HTTP	533	HTTP/1.1 404 Not Found (text/html)
468	58.394426	172.16.22.1	192.168.254.254	HTTP	433	GET / HTTP/1.1
470	58.397330	192.168.254.254	172.16.22.1	HTTP	199	HTTP/1.1 304 Not Modified
478	58.408940	172.16.22.1	192.168.254.254	HTTP	287	GET /favicon.ico HTTP/1.1
480	58.412005	192.168.254.254	172.16.22.1	HTTP	533	HTTP/1.1 404 Not Found (text/html)
773	96.516228	172.16.22.1	192.168.254.254	HTTP	357	GET /page2.html/ HTTP/1.1
775	96.519378	192.168.254.254	172.16.22.1	HTTP	533	HTTP/1.1 404 Not Found (text/html)
813	98.940508	172.16.22.1	192.168.254.254	HTTP	356	GET /page2.html HTTP/1.1
815	98.943628	192.168.254.254	172.16.22.1	HTTP	530	HTTP/1.1 200 OK (text/html)
833	101.841830	172.16.22.1	192.168.254.254	HTTP	443	GET /page2.html HTTP/1.1
835	101.844317	192.168.254.254	172.16.22.1	HTTP	199	HTTP/1.1 304 Not Modified

Result from Wireshark HTTP packet capture for Part A shown above. Red box "1" above shows first HTTP session between web browser (client) and web server (host). First line shows client sending request to host to access the default webpage (specified in first URL mentioned previously). Second line shows host responding to client by sending requested webpage to client to be displayed on its web browser. Red box "2" above shows second HTTP session between client and host. This session shows what occurs when the same requested webpage from the client is asked for a second time to the host. Red box "3" and "4" above are similar to the previous two red boxes. The only difference is that the client is deliberately requesting a specific webpage (specified in second URL mentioned previously). Other HTTP packet captures above are the result from mistyped URLs.

- b) Fill in the following HTTP session table based on your first captured HTTP session data.

Characteristic	Answers
Web browser IP address	172.16.22.1 (my computer – shown in red box below)
Web server IP address	192.168.254.254 (Eagle server – shown in red box below)
Transport layer protocol (UDP/TCP)	TCP (Shown in orange box below)
Web browser port number	53331 (randomly chosen – shown in yellow box below)
Web server port number	80 (specifically chosen for HTTP – shown in yellow box)

See screenshot below:

```
> Frame 424: 346 bytes on wire (2768 bits), 346 bytes captured (2768 bits) on interface \Device\NPF_{F46C336C-16C4-40CE-BF3A-ABEE5B4E3CDE}, id 0
> Ethernet II, Src: Dell_a2:ce:bd (d4:be:d9:a2:ce:bd), Dst: Cisco_6b:d2:88 (00:19:56:6b:d2:88)
> Internet Protocol Version 4, Src: 172.16.22.1, Dst: 192.168.254.254
> Transmission Control Protocol, Src Port: 53331, Dst Port: 80, Seq: 1, Ack: 1, Len: 292
> Hypertext Transfer Protocol
```

Screenshot taken from first captured HTTP packet in Wireshark for Part A. Various coloured boxes represent the answers to the characteristics asked in the table above.

- Which computer initiated the HTTP session, and what are the handshake procedures involved?
 - The client (web browser) initiated the HTTP session to the host (web server). The client sends a request (using "GET" and HTTP) to the host to request a specific webpage to be displayed on the client's web browser. The host responds (using HTTP and "OK") with the requested webpage by sending the corresponding HTML file to be displayed on the client's web browser. To prevent the HTTP session between the client and host from closing, the client signals a "Keep-Alive" signal until the HTML file is delivered to the client. See red box "1" in screenshot above and screenshot below:

```
▼ Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Accept: text/html, application/xhtml+xml, image/jxr, */*\r\n
    Accept-Language: en-US,en-CA;q=0.7,en;q=0.3\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko\r\n
    Accept-Encoding: gzip, deflate\r\n
    Host: eagle-server.example.com\r\n
    Connection: Keep-Alive\r\n
  \r\n
  [Full request URI: http://eagle-server.example.com/]
  [HTTP request 1/1]
  [Response in frame: 426]
```

Screenshot taken from first captured HTTP packet in Wireshark for Part A. As shown in the red box above, the client sends a "Keep-Alive" signal to the host to ensure the client receives the desired HTML file to be displayed on its web browser.

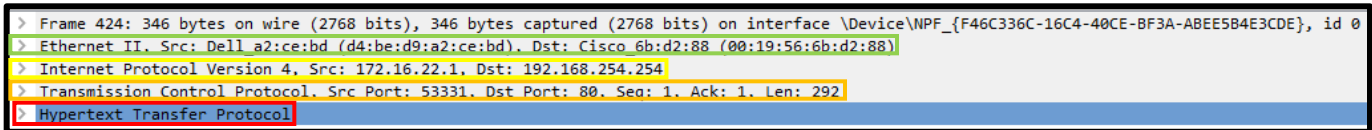
- Which computer initially signaled an end to the HTTP session, and how?
 - The host (web server) was the first computer who initially signaled an end to the HTTP session to the client (web browser). The host was able to do so by signalling a "close" signal to the client. See screenshot below:

```
▼ Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Date: Fri, 24 Sep 2021 03:24:32 GMT\r\n
    Server: Apache/2.0.52 (Fedora)\r\n
    Last-Modified: Fri, 26 Jan 2007 06:20:11 GMT\r\n
    ETag: "150b52-b8-851e5cc0"\r\n
    Accept-Ranges: bytes\r\n
  > Content-Length: 184\r\n
    Connection: close\r\n
    Content-Type: text/html; charset=UTF-8\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.003410000 seconds]
  [Request in frame: 424]
  [Request URI: http://eagle-server.example.com/]
  File Data: 184 bytes
```

Screenshot taken from second captured HTTP packet in Wireshark for Part A. As shown in the red box above, the host sends a "close" signal to the client to say that the desired HTML file has been sent to the client and that the HTTP session will now close.

- c) Highlight the first line of the HTTP protocol, a GET request from the web browser. Examine the layered protocols. Which protocols are carried (encapsulated) inside the TCP segment?

- The HTTP protocol is the only protocol encapsulated inside the TCP segment. See screenshot below:



Screenshot taken from first captured HTTP packet in Wireshark for Part A. The four highlighted boxes above represent the four layers of the TCP/IP OSI model. The red box is the application layer, the orange box is the transport layer, the yellow box is the internet layer, and the light green box is the network access layer. As shown above, we can see that the HTTP protocol is the only protocol encapsulated inside the TCP segment.

Complete the following table using the information from the protocol:

Characteristic	Answers
Protocol version	1.1 (Shown in red box below)
Request method	GET (Shown in orange box below)
Request URI	http://eagle-server.example.com/ (Shown in yellow box below)
Language	en-US and en-CA (Shown in light green box below)

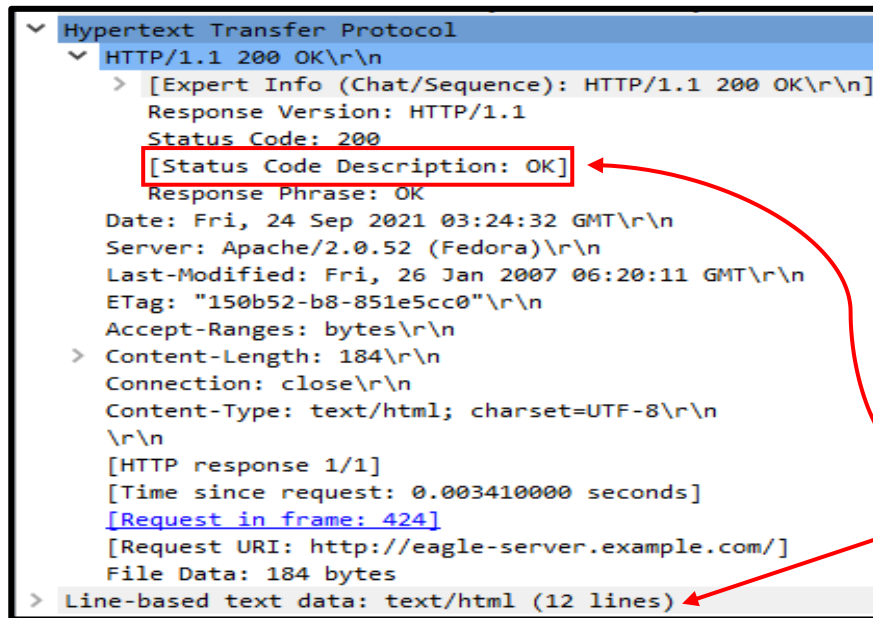
See screenshot below:



Screenshot taken from first captured HTTP packet in Wireshark for Part A. Various coloured boxes represent the answers to the characteristics asked in the table above.

- What is the web server response to the web client GET request?
 - The host (web server) response to the client's (web browser) GET request is the corresponding HTML file to be displayed on the client's web browser. The host also responds with an "Ok" message. See red box "1" in screenshot under question 2a above.

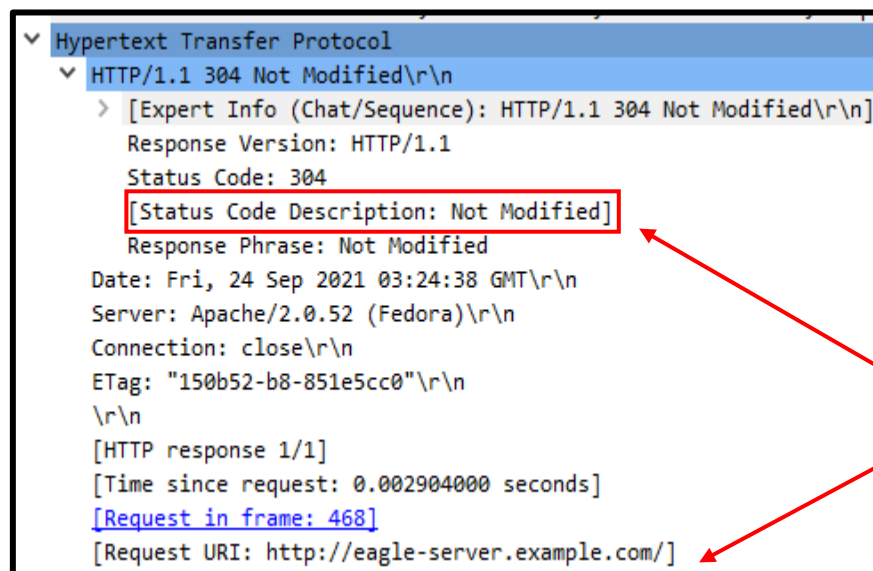
- d) Scroll until the second HTTP session, where the refresh happened. What is the response from the server on this request? Examine the details of the GET request and find out what additional instruction has been given to the web server.
- The response from the host (web server) to the client (web browser) is a "Not Modified" message as seen in red box "2" shown in screenshot under question 2a above. See additional screenshots below:



```
▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 200 OK\r\n
    > [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
      Date: Fri, 24 Sep 2021 03:24:32 GMT\r\n
      Server: Apache/2.0.52 (Fedora)\r\n
      Last-Modified: Fri, 26 Jan 2007 06:20:11 GMT\r\n
      ETag: "150b52-b8-851e5cc0"\r\n
      Accept-Ranges: bytes\r\n
    > Content-Length: 184\r\n
      Connection: close\r\n
      Content-Type: text/html; charset=UTF-8\r\n
      \r\n
      [HTTP response 1/1]
      [Time since request: 0.003410000 seconds]
      [Request in frame: 424]
      [Request URI: http://eagle-server.example.com/]
      File Data: 184 bytes
    > Line-based text data: text/html (12 lines)
```

Host responds with requested HTML file along with a "Ok" message.

Screenshot taken from second captured HTTP packet in Wireshark for Part A (GET request response in first session). Screenshot above shows the contents of the first GET request response from the host to client.



```
▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 304 Not Modified\r\n
    > [Expert Info (Chat/Sequence): HTTP/1.1 304 Not Modified\r\n]
      Response Version: HTTP/1.1
      Status Code: 304
      [Status Code Description: Not Modified]
      Response Phrase: Not Modified
      Date: Fri, 24 Sep 2021 03:24:38 GMT\r\n
      Server: Apache/2.0.52 (Fedora)\r\n
      Connection: close\r\n
      ETag: "150b52-b8-851e5cc0"\r\n
      \r\n
      [HTTP response 1/1]
      [Time since request: 0.002904000 seconds]
      [Request in frame: 468]
      [Request URI: http://eagle-server.example.com/]
```

Host responds with no requested HTML file along with a "Not Modified" message.

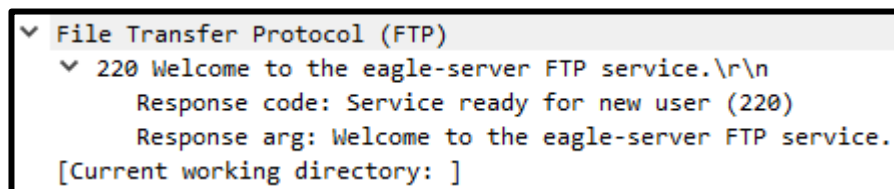
Screenshot taken from sixth captured HTTP packet in Wireshark for Part A (GET request response in second session). Screenshot above shows the contents of the second GET request response from the host to client.

Part B: Capture and analyze FTP communication between host and a web server

The web browser can be used to communicate with more than just the HTTP server. In part B, the web browser and a command-line FTP utility are used to download data from an FTP server.

3. Start a new Wireshark capture, and then start a command-line FTP client.
 - a) Enter the following command. Use user ID of *anonymous*, and when prompted for a password, press <Enter>.
`ftp eagle-server.example.com`
 - b) Enter `dir` to display the current directory contents.
 - c) Move into directory `pub/eagle_labs/eagle1/chapter2`, download a file, and exit.
 - d) Close command line window, and stop Wireshark capture. Save the capture.
4. Start a web browser on host computer, and start a new Wireshark capture again.
 - a) Enter the URL <ftp://eagle-server.example.com>. Now the web browser logged in to the FTP server as user anonymous.
 - b) Using the browser, go down the directories until the URL path is `pub/eagle1labs/eagle1/chapter2`. Double-click the file `ftptoeagle-server.pcap` and save it.
 - c) Close the web browser, when finished. Stop Wireshark and save the captures.
5. Analyze the captured data of FTP via web browser (show screen shot of your captured data whenever necessary).
 - a) Select the first FTP protocol transmission, and examine the FTP protocol. What is the meaning of FTP server response 220?

- The meaning of the host (FTP server) response "220" is to signal the client (web browser → FTP client) that the FTP service is ready for the user. See screenshot below:



Screenshot taken from first captured FTP packet in Wireshark for Part B. Screenshot shows the contents of the first captured FTP packet in Wireshark and the meaning of the host response "220".

b) When FTP issued a "Response: 331 Please specify the password.", what did the web browser reply?

- The client (web browser → FTP client) replied with the requested password from the host (FTP server). In this case, the client replied with the default password (none) to the host's request. See screenshot below:

```
File Transfer Protocol (FTP)
  PASS IEUser@\r\n
    Request command: PASS
    Request arg: IEUser@
    [Current working directory: ]
```

Screenshot taken from fourth captured FTP packet in Wireshark for Part B. Screenshot shows the contents of the fourth captured FTP packet in Wireshark.

c) Which port number does the FTP client use to connect to the FTP server port 21?

- The port number the client (web browser → FTP client) uses to connect to the host (FTP server) port 21 is port 53377. This port number the client uses to connect to the host is randomly assigned to itself. See screenshot below:

```
> Frame 141: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface \Device\NPF_{F46C336C-16C4-40CE-BF3A-ABEE5B4E3CDE}, id 0
> Ethernet II, Src: Cisco_6b:d2:88 (00:19:56:6b:d2:88), Dst: Dell_a2:ce:bd (d4:be:d9:a2:ce:bd)
> Internet Protocol Version 4, Src: 192.168.254.254, Dst: 172.16.22.1
> Transmission Control Protocol, Src Port: 21, Dst Port: 53377 Seq: 1, Ack: 1, Len: 46
> File Transfer Protocol (FTP)
  [Current working directory: ]
```

Screenshot taken from first captured FTP packet in Wireshark for Part B. Screenshot shows the contents of the first captured FTP packet in Wireshark and red box shows the port number assigned to the client used to connect to the host.

6. Analyze the captured data of FTP via command line and observe the similar instructions and transactions.

- See screenshots and comments below:

No.	Time	Source	Destination	Protocol	Length	Info
141	22.020601	192.168.254.254	172.16.22.1	FTP	100	Response: 220 Welcome to the eagle-server FTP service.
143	22.020680	172.16.22.1	192.168.254.254	FTP	70	Request: USER anonymous
145	22.021553	192.168.254.254	172.16.22.1	FTP	88	Response: 331 Please specify the password.
147	22.021623	172.16.22.1	192.168.254.254	FTP	68	Request: PASS IEUser@
148	22.025140	192.168.254.254	172.16.22.1	FTP	77	Response: 230 Login successful.
150	22.025225	172.16.22.1	192.168.254.254	FTP	61	Request: CWD /
151	22.026274	192.168.254.254	172.16.22.1	FTP	91	Response: 250 Directory successfully changed.
153	22.026419	172.16.22.1	192.168.254.254	FTP	62	Request: TYPE A
154	22.027355	192.168.254.254	172.16.22.1	FTP	84	Response: 200 Switching to ASCII mode.
156	22.027526	172.16.22.1	192.168.254.254	FTP	60	Request: PASV
157	22.028619	192.168.254.254	172.16.22.1	FTP	107	Response: 227 Entering Passive Mode (192,168,254,254,182,186)
162	22.029641	172.16.22.1	192.168.254.254	FTP	60	Request: LIST
163	22.030775	192.168.254.254	172.16.22.1	FTP	93	Response: 150 Here comes the directory listing.
166	22.031129	192.168.254.254	172.16.22.1	FTP	78	Response: 226 Directory send OK.

Screenshot taken from first captured FTP session in Wireshark for Part B. Screenshot shows the complete exchange between the client and the host when the client attempts to retrieve a file on the host (client is using web browser).

No.	Time	Source	Destination	Protocol	Length	Info
160	37.694874	192.168.254.254	172.16.22.1	FTP	100	Response: 220 Welcome to the eagle-server FTP service.
161	37.698060	172.16.22.1	192.168.254.254	FTP	68	Request: OPTS UTF8 ON
163	37.698905	192.168.254.254	172.16.22.1	FTP	92	Response: 530 Please login with USER and PASS.
222	49.513596	172.16.22.1	192.168.254.254	FTP	70	Request: USER anonymous
223	49.514669	192.168.254.254	172.16.22.1	FTP	88	Response: 331 Please specify the password.
235	52.489876	172.16.22.1	192.168.254.254	FTP	61	Request: PASS
236	52.492439	192.168.254.254	172.16.22.1	FTP	77	Response: 230 Login successful.
261	57.945698	172.16.22.1	192.168.254.254	FTP	80	Request: PORT 172,16,22,1,208,120
262	57.946913	192.168.254.254	172.16.22.1	FTP	105	Response: 200 PORT command successful. Consider using PASV.
263	57.950438	172.16.22.1	192.168.254.254	FTP	60	Request: LIST
267	57.952456	192.168.254.254	172.16.22.1	FTP	93	Response: 150 Here comes the directory listing.
269	57.953025	192.168.254.254	172.16.22.1	FTP	78	Response: 226 Directory send OK.

Screenshot taken from first captured FTP session in Wireshark for Part B. Screenshot shows the complete exchange between the client and the host when the client attempts to retrieve a file on the host (client is using command prompt).

Between the two screenshots, we can see a similarity. Regardless of the way the client accesses the host, the client needs to specify a username and password, navigate to a desired directory to retrieve a desired file and wait for the host to send the desired file to the client's root directory.

7. What is/are the transfer mode(s) used by the two FTP sessions?

- The transfer mode used by the two FTP sessions is active mode. The host initiates the data transfer on port 21 and the client receives the data transfer on port 53377 (port assigned when using web browser) or port 53366 (port assigned when using command prompt). See screenshots below:

```
> Frame 166: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface \Device\NPF_{F46C336C-16C4-40CE-BF3A-ABEE5B4E3CDE}, id 0
> Ethernet II, Src: Cisco_6b:d2:88 (00:19:56:6b:d2:88), Dst: Dell_a2:ce:bd (d4:be:d9:a2:ce:bd)
> Internet Protocol Version 4, Src: 192.168.254.254, Dst: 172.16.22.1
> Transmission Control Protocol, Src Port: 21, Dst Port: 53377, Seq: 263, Ack: 58, Len: 24
> File Transfer Protocol (FTP)
[Current working directory: /]
```

Screenshot taken from 14th captured FTP packet in Wireshark for Part B. Red box shows that the host initiates the data transfer on port 21 and that the client receives the data transfer on port 53377 (client is using web browser).

```
> Frame 269: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface \Device\NPF_{F46C336C-16C4-40CE-BF3A-ABEE5B4E3CDE}, id 0
> Ethernet II, Src: Cisco_6b:d2:88 (00:19:56:6b:d2:88), Dst: Dell_a2:ce:bd (d4:be:d9:a2:ce:bd)
> Internet Protocol Version 4, Src: 192.168.254.254, Dst: 172.16.22.1
> Transmission Control Protocol, Src Port: 21, Dst Port: 53366, Seq: 232, Ack: 70, Len: 24
> File Transfer Protocol (FTP)
[Current working directory: ]
```

Screenshot taken from 12th captured FTP packet in Wireshark for Part B. Red box shows that the host initiates the data transfer on port 21 and that the client receives the data transfer on port 53366 (client is using command prompt).

Discussion:

- This lab was completed without that many issues compared to previous labs. First, we observed a typical exchange between a web server (host) and a web browser (client). The client would initiate a request to the host to be able to retrieve a specific webpage on its web browser. The host would reply with the requested webpage in the format of HTML (HTML file). Other parameters were observed during this exchange, such as the responses from the client's web page requests. Basic networking details, such as source and destination IP address, source and destination ports, were observed as well. Secondly, we observed a typical exchange between an FTP server (host) and an FTP client (client). We observed this exchange two different ways: 1) using a web browser as an FTP client and 2) using the command prompt in Windows as an FTP client. Similar to the previous exchange, the client would similarly initiate a request to the host to retrieve a specific file. The host would reply with the requested file and would save it to the client's root directory. The same various parameters, such as host responses, were observed during this exchange (as was done in the previous exchange). Basic networking details were observed as well (same as in the previous exchange). For both of these exchanges (HTTP and FTP), a Wireshark packet capture was running at the same time in order to analyze these fine details later on. Similar to the previous lab(s), the insecurities are visible in this lab. For example, during the HTTP exchange, we were able to see the very fine details of the host's responses of web pages to the client. For the FTP exchange, we were also able to see the similar very fine details. As was done before in previous labs, we can conclude that these two exchanges have a serious flaw that put the end users (and maybe others as well) at risk of data breaches. As a result, more secure ways exist to perform these same exchanges (such as HTTPS and FTPS).

Conclusion:

- Successfully configured client computer to capture application layer protocols.
- Successfully configured client computer to capture transport layer protocols.
- Successfully captured and analyzed HTTP communication between client and host.
- Successfully captured and analyzed FTP communication between client and host.
- Successfully observed transport layer protocol TCP establish and manage communication channels with HTTP and FTP connections.