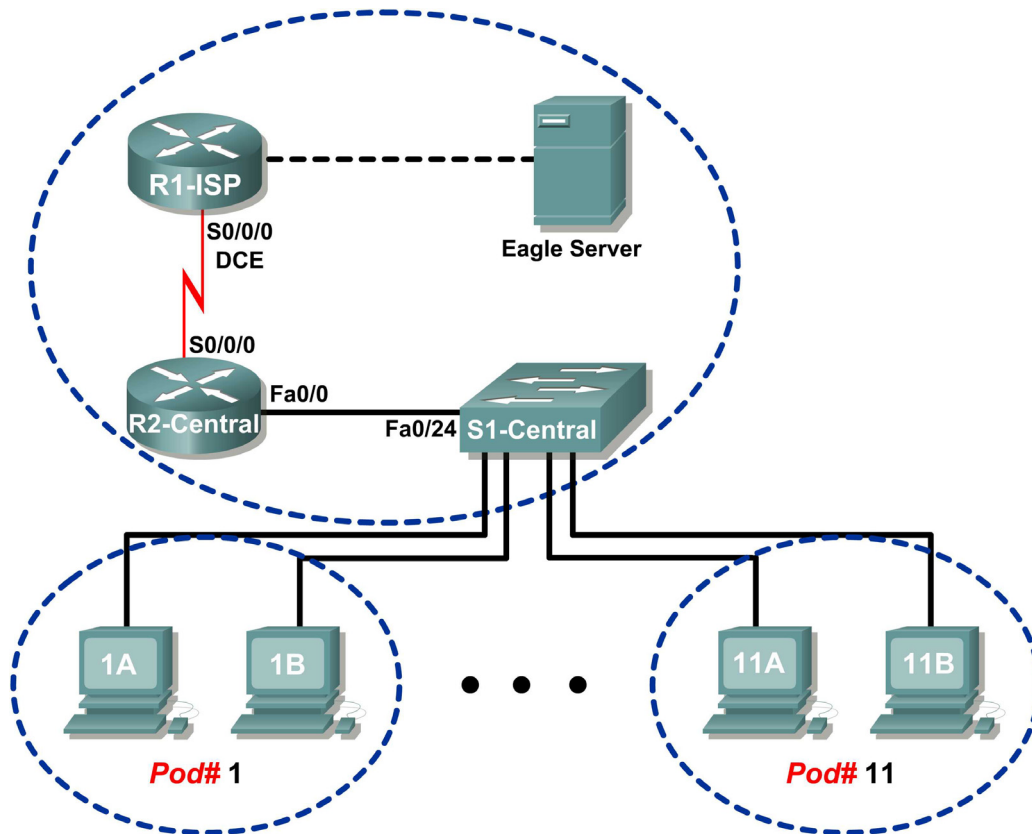


Lab 4.5.3: Application and Transport Layer Protocols Examination

Topology Diagram



Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
R1-ISP	S0/0/0	10.10.10.6	255.255.255.252	N/A
	Fa0/0	192.168.254.253	255.255.255.0	N/A
R2-Central	S0/0/0	10.10.10.5	255.255.255.252	N/A
	Fa0/0	172.16.255.254	255.255.0.0	N/A
Eagle Server	N/A	192.168.254.254	255.255.255.0	192.168.254.253
	N/A	172.31.24.254	255.255.255.0	N/A
hostPod#A	N/A	172.16.Pod#.1	255.255.0.0	172.16.255.254
hostPod#B	N/A	172.16.Pod#.2	255.255.0.0	172.16.255.254
S1-Central	N/A	172.16.254.1	255.255.0.0	172.16.255.254

Learning Objectives

Upon completion of this lab, you will be able to:

- Configure the host computer to capture Application layer protocols.
- Capture and analyze HTTP communication between the pod host computer and a web server.
- Capture and analyze FTP communication between the pod host computer and an FTP server.
- Observe TCP establish and manage communication channels with HTTP and FTP connections

Background

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 there is a different server 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 or UDP. This lab will examine two popular Application Layer protocols, HTTP and FTP, and how Transport Layer protocols TCP and UDP manage the communication channel. Also examined are popular client requests and corresponding server responses.

Scenario

In this lab, you will use client applications to connect to eagle-server network services. You will monitor the communication with Wireshark and analyze the captured packets.

A web browser such as Internet Explorer or Firefox will be used to connect to the eagle-server network service. Eagle-server has several network services preconfigured, such as HTTP, waiting to respond to client requests.

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

Task 1: Configure the Pod Host Computer to Capture Application Layer Protocols.

The lab should be configured as shown in the Topology Diagram and logical address table. If it is not, ask the instructor for assistance before proceeding.

Step 1: Download and install wireshark.



Figure 1. FTP Download for Wireshark

If Wireshark is not installed on the pod host computer, it can be downloaded from eagle-server.example.com. See Figure 1. The download URL is ftp://eagle-server.example.com/pub/eagle_labs/eagle1/chapter3/.

1. Right-click the wireshark filename, then save the file to the host pod computer.
2. When the file has downloaded, double-click the filename and install Wireshark with the default settings.

Step 2: Start Wireshark and configure the Capture Interface.

1. Start Wireshark from **Start > All Programs > Wireshark > Wireshark**.
2. When the opening screen appears, set the correct Capture Interface. The interface with the IP address of the pod host computer is the correct interface. See Figure 2.

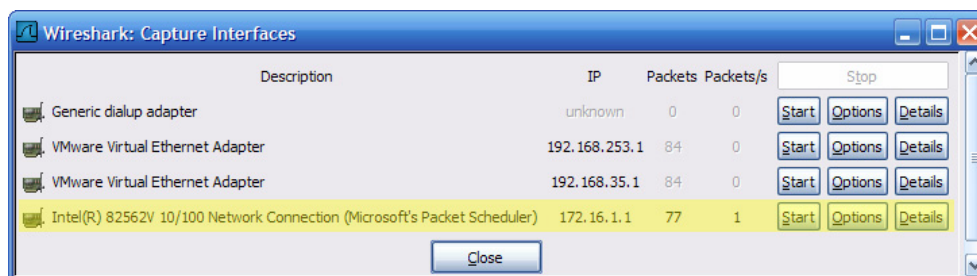


Figure 2. Wireshark Interface Capture Screen

Wireshark can be started by clicking the interface **Start** button. Thereafter, the interface is used as the default and does not need to be changed.

Wireshark should begin to log data.

3. Stop Wireshark for the moment. Wireshark will be used in upcoming tasks.

Task 2: Capture and Analyze HTTP Communication Between the Pod Host Computer 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. HTTP version 1.1 is defined in RFC 2616, dated 1999. This part of the lab will demonstrate how sessions between multiple web clients and the web server are kept separate.

Step 1: Start Wireshark captures.

Start a Wireshark capture. Wireshark will display captures based on packet type.

Step 2: Start the pod host web browser.

1. Using a web browser such as Internet Explorer or Firefox, connect to URL <http://eagle-server.example.com>. A web page similar to Figure 3 will be displayed. Do not close this web browser until instructed to do so.

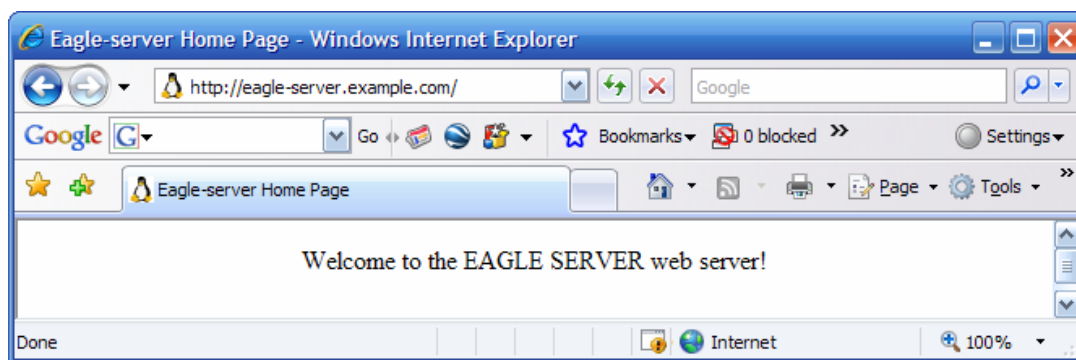


Figure 3. Web Browser Connected to Web Server

2. Click the web browser **Refresh** button. There should be no change to the display in the web client.
3. Open a second web browser, and connect to URL <http://eagle-server.example.com/page2.html>. This will display a different web page.

Do not close either browser until Wireshark capture is stopped.

Step 3: Stop Wireshark captures and analyze the captured data.

1. Stop Wireshark captures.
2. Close the web browsers.

The resulting Wireshark data will be displayed. There were actually at least three HTTP sessions created in Step 2. The first HTTP session started with a connection to <http://eagle-server.example.com>. The second session occurred with a refresh action. The third session occurred when the second web browser accessed <http://eagle-server.example.com/page2.html>.

No. -	Time	Source	Destination	Protocol	Info
10	10.168217	172.16.1.2	192.168.254.254	TCP	1056 > http [SYN] Seq=0 Len=0 MSS=1460
11	10.170734	192.168.254.254	172.16.1.2	TCP	http > 1056 [SYN, ACK] Seq=0 Ack=1 win=5840 Len=0 MSS=1460
12	10.170767	172.16.1.2	192.168.254.254	TCP	1056 > http [ACK] Seq=1 Ack=1 win=64240 Len=0
13	10.171086	172.16.1.2	192.168.254.254	HTTP	GET / HTTP/1.1
14	10.171625	192.168.254.254	172.16.1.2	TCP	http > 1056 [ACK] Seq=1 Ack=208 win=6432 Len=0
15	10.172518	192.168.254.254	172.16.1.2	HTTP	HTTP/1.1 200 OK (text/html)
16	10.172540	192.168.254.254	172.16.1.2	TCP	http > 1056 [FIN, ACK] Seq=448 Ack=208 win=6432 Len=0
17	10.172567	172.16.1.2	192.168.254.254	TCP	1056 > http [ACK] Seq=208 Ack=449 win=63793 Len=0
18	10.174196	172.16.1.2	192.168.254.254	TCP	1056 > http [FIN, ACK] Seq=208 Ack=449 win=63793 Len=0
19	10.174661	192.168.254.254	172.16.1.2	TCP	http > 1056 [ACK] Seq=449 Ack=209 win=6432 Len=0

Figure 4. Captured HTTP Session

A sample captured HTTP session is shown in Figure 4. Before HTTP can begin, the TCP session must be created. This is seen in the first three session lines, numbers 10, 11, and 12. Use your capture or similar Wireshark output to answer the following questions:

3. Fill in the following table from the information presented in the HTTP session:

Web browser IP address	
Web server IP address	
Transport layer protocol (UDP/TCP)	
Web browser port number	
Web server port number	

4. Which computer initiated the HTTP session, and how?

5. Which computer initially signaled an end to the HTTP session, and how?

6. Highlight the first line of the HTTP protocol, a **GET** request from the web browser. In Figure 4 above, the **GET** request is on line 13. Move into the second (middle) Wireshark window to examine the layered protocols. If necessary, expand the fields.

7. Which protocol is carried (encapsulated) inside the TCP segment?

8. Expand the last protocol record, and any subfields. This is the actual information sent to the web server. Complete the following table using information from the protocol.

Protocol Version	
Request Method	
* Request URI	
Language	

- * Request URI is the path to the requested document. In the first browser, the path is the root directory of the web server. Although no page was requested, some web servers are configured to display a default file if one is available.

The web server responds with the next HTTP packet. In Figure 4, this is on line 15. A response to the web browser is possible because the web server (1) understands the type of request and (2) has a file to return. Crackers sometimes send unknown or garbled requests to web servers in an attempt to stop the server or gain access to the server command line. Also, a request for an unknown web page will result in an error message.

9. Highlight the web server response, and then move into the second (middle) window. Open all collapsed sub-fields of HTTP. Notice the information returned from the server. In this reply, there are only a few lines of text (web server responses can contain thousands or millions of bytes). The web browser understands and correctly formats the data in the browser window. .
10. What is the web server response to the web client **GET** request?

-
11. What does this response mean?
-

12. Scroll down the top window of Wireshark until the second HTTP session, refresh, is visible. A sample capture is shown in Figure 5.

21	12.487941	172.16.1.2	192.168.254.254	TCP	1057 > http [SYN] Seq=0 Len=0 MSS=1460
22	12.488485	192.168.254.254	172.16.1.2	TCP	http > 1057 [SYN, ACK] Seq=0 Ack=1 win=5840 Len=0 MSS=1460
23	12.488526	172.16.1.2	192.168.254.254	TCP	1057 > http [ACK] Seq=1 Ack=1 win=64240 Len=0
24	12.488864	172.16.1.2	192.168.254.254	HTTP	GET / HTTP/1.1
25	12.489370	192.168.254.254	172.16.1.2	TCP	http > 1057 [ACK] Seq=1 Ack=294 win=6432 Len=0
26	12.489927	192.168.254.254	172.16.1.2	HTTP	HTTP/1.1 304 Not Modified
27	12.489953	192.168.254.254	172.16.1.2	TCP	http > 1057 [FIN, ACK] Seq=145 Ack=294 win=6432 Len=0
28	12.489989	172.16.1.2	192.168.254.254	TCP	1057 > http [ACK] Seq=294 Ack=146 win=64096 Len=0
29	12.490345	172.16.1.2	192.168.254.254	TCP	1057 > http [FIN, ACK] Seq=294 Ack=146 win=64096 Len=0
30	12.490705	192.168.254.254	172.16.1.2	TCP	http > 1057 [ACK] Seq=146 Ack=295 win=6432 Len=0

Figure 5. Captured HTTP Session for Refresh

The significance of the refresh action is in the server response, 304 Not Modified. With a single packet returned for both the initial **GET** request and refresh, the bandwidth used is minimal. However, for an initial response that contains millions of bytes, a single reply packet can save significant bandwidth.

Because this web page was saved in the web client's cache, the **GET** request contained the following additional instructions to the web server:

```
If-modified-since: Fri, 26 Jan 2007 06:19:33 GMT\r\n
If-None-Match: "98072-b8-82da8740"\r\n <- page tag number (ETAG)
```

13. What is the ETAG response from the web server?
-

Task 3: Capture and Analyze FTP Communication Between the Pod Host Computer and a Web Server.

The Application layer protocol FTP has undergone significant revision since it first appeared in RFC 114, in 1971. FTP version 5.1 is defined in RFC 959, dated October, 1985.

The familiar web browser can be used to communicate with more than just the HTTP server. In this task, the web browser and a command line FTP utility will be used to download data from an FTP server.

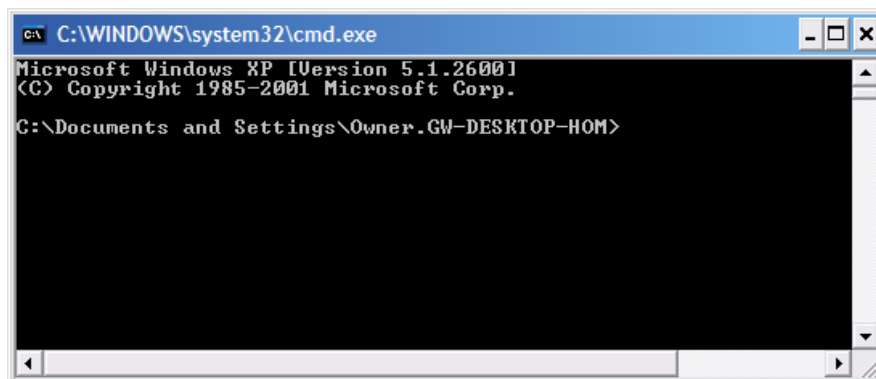


Figure 6. Windows Command Line Screen

In preparation for this task, open a command line on the host pod computer. This can be accomplished by clicking **Start > Run**, then typing **CMD** and clicking **OK**. A screen similar to Figure 6 will be displayed.

Step 1: Start Wireshark captures.

If necessary, refer to Task 1, Step 2, to open Wireshark.

Step 2: Start the pod host command line FTP client.

1. Start a pod host computer FTP session with the FTP server, using the Windows FTP client utility. To authenticate, use userid **anonymous**. In response to the password prompt, press **<ENTER>**.

```
>ftp eagle-server.example.com
Connected to eagle-server.example.com.
220 Welcome to the eagle-server FTP service.
User (eagle-server.example.com:(none)): anonymous
331 Please specify the password.
Password: <ENTER>
230 Login successful.
```

2. The FTP client prompt is **ftp>**. This means that the FTP client is waiting for a command to send to the FTP server. To view a list of FTP client commands, type **help <ENTER>**:

```
ftp> help
Commands may be abbreviated.  Commands are:

!           delete          literal      prompt      send
?           debug           ls           put          status
append     dir                   mdelete     pwd          trace
ascii      disconnect      mdir        quit         type
bell       get              mget        quote        user
binary     glob             mkdir       recv         verbose
bye        hash             mls         remotehelp
cd         help           mput        rename
close     lcd             open        rmdir
```

Unfortunately, the large number of FTP client commands makes using the command line utility difficult for a novice. We will only use a few commands for Wireshark evaluation.

3. Type the command **dir** to display the current directory contents:

```
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    3 0          0          4096 Jan 12 04:32 pub
```

The FTP client is at the root directory of the FTP server. This is not the real root directory of the server—only the highest point that user **anonymous** can access. User **anonymous** has been placed into a root jail, prohibiting access outside of the current directory.

4. Subdirectories can be traversed, however, and files transferred to the pod host computer. Move into directory `pub/eagle_labs/eagle1/chapter2`, download a file, and exit.

```
ftp> cd pub/eagle_labs/eagle1/chapter2
250 Directory successfully changed.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0 100      5853 Jan 12 04:26 ftptoeagle-server.pcap
-rw-r--r--    1 0 100      4493 Jan 12 04:27 http to eagle-server.pcap
-rw-r--r--    1 0 100      1486 Jan 12 04:27 ping to 192.168.254.254.pcap
-rw-r--r--    1 0 100    15163750 Jan 12 04:30 wireshark-setup-0.99.4.exe
226 Directory send OK.
ftp: 333 bytes received in 0.04Seconds 8.12Kbytes/sec.
ftp> get "ftptoeagle-server.pcap"
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for ftptoeagle-server.pcap (5853 bytes).
226 File send OK.
ftp: 5853 bytes received in 0.34Seconds 17.21Kbytes/sec.
ftp> quit
221 Goodbye.
```

5. Close the command line window with the exit command.
6. Stop Wireshark captures, and save the captures as `FTP_Command_Line_Client`.

Step 3: Start the pod host web browser.

1. Start Wireshark captures again.

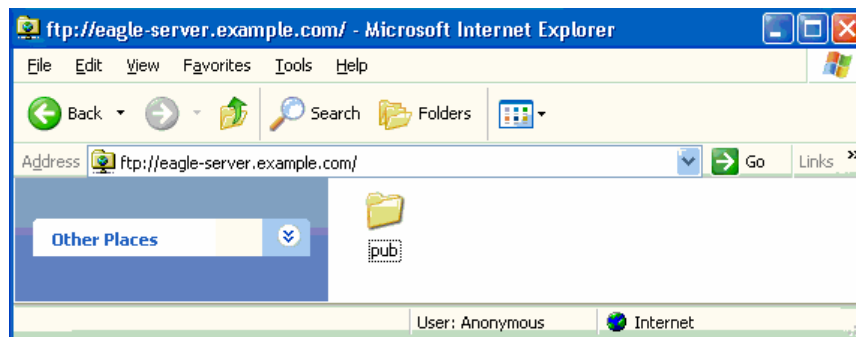


Figure 7. Web Browser Used as an FTP Client

2. Open a web browser as shown in Figure 7, and type in URL <ftp://eagle-server.example.com>. A browser window opens with the pub directory displayed. Also, the web browser logged into the FTP server as user Anonymous as shown on the bottom of the screen capture.
3. Using the browser, go down the directories until the URL path is `pub/eagle-labs/eagle1/chapter2`. Double-click the file `ftptoeagle-server.pcap` and save the file.
4. When finished, close the web browser.
5. Stop Wireshark captures, and save the captures as `FTP_Web_Browser_Client`.

Step 4: Stop Wireshark captures and analyze the captured data.

1. If not already opened, open the Wireshark capture `FTP_Web_Browser_Client`.
2. On the top Wireshark window, select the FTP capture that is the first FTP protocol transmission, Response: 220. In Figure 8, this is line 23.

No.	Time	Source	Destination	Protocol	Info
12	16.276555	172.16.1.2	192.168.254.254	DNS	Standard query A eagle-server.example.com
13	16.277284	192.168.254.254	172.16.1.2	DNS	Standard query response A 192.168.254.254
14	16.278059	172.16.1.2	192.168.254.254	TCP	1073 > ftp [SYN] Seq=0 Len=0 MSS=1460
15	16.278540	192.168.254.254	172.16.1.2	TCP	ftp > 1073 [SYN, ACK] Seq=0 Ack=1 win=5840 Len=0 MSS=1460
16	16.278575	172.16.1.2	192.168.254.254	TCP	1073 > ftp [ACK] Seq=1 Ack=1 win=64240 Len=0
23	26.281472	192.168.254.254	172.16.1.2	FTP	Response: 220 welcome to the eagle-server FTP service.
24	26.281672	172.16.1.2	192.168.254.254	FTP	Request: USER anonymous
25	26.282120	192.168.254.254	172.16.1.2	TCP	ftp > 1073 [ACK] Seq=47 Ack=17 win=5840 Len=0
26	26.282137	192.168.254.254	172.16.1.2	FTP	Response: 331 Please specify the password.
27	26.282201	172.16.1.2	192.168.254.254	FTP	Request: PASS iUser@
28	26.283451	192.168.254.254	172.16.1.2	FTP	Response: 230 Login successful.
29	26.313423	172.16.1.2	192.168.254.254	FTP	Request: opts utf8 on
30	26.313959	192.168.254.254	172.16.1.2	FTP	Response: 501 option not understood.
31	26.314042	172.16.1.2	192.168.254.254	FTP	Request: syst
32	26.314493	192.168.254.254	172.16.1.2	FTP	Response: 215 UNIX Type: L8
33	26.314595	172.16.1.2	192.168.254.254	FTP	Request: site help
34	26.315028	192.168.254.254	172.16.1.2	FTP	Response: 550 Permission denied.
35	26.315113	172.16.1.2	192.168.254.254	FTP	Request: PWD
36	26.315566	192.168.254.254	172.16.1.2	FTP	Response: 257 "/"
37	26.352350	172.16.1.2	192.168.254.254	FTP	Request: noop
38	26.352821	192.168.254.254	172.16.1.2	FTP	Response: 200 NOOP ok.
39	26.482680	172.16.1.2	192.168.254.254	FTP	Request: CWD /
40	26.483243	192.168.254.254	172.16.1.2	FTP	Response: 250 Directory successfully changed.
41	26.484334	172.16.1.2	192.168.254.254	FTP	Request: TYPE A
42	26.484824	192.168.254.254	172.16.1.2	FTP	Response: 200 Switching to ASCII mode.
43	26.485292	172.16.1.2	192.168.254.254	FTP	Request: PORT 172,16,1,2,4,50
44	26.485800	192.168.254.254	172.16.1.2	FTP	Response: 200 PORT command successful. Consider using PASV.
45	26.485892	172.16.1.2	192.168.254.254	FTP	Request: LIST
46	26.486503	192.168.254.254	172.16.1.2	TCP	ftp-data > 1074 [SYN] Seq=0 Len=0 MSS=1460 TSV=12998374 TSER=0 WS=2
47	26.486558	172.16.1.2	192.168.254.254	TCP	1074 > ftp-data [SYN, ACK] Seq=0 Ack=1 win=64240 Len=0 MSS=1460 WS=0 TSV=0 TSER=
48	26.486948	192.168.254.254	172.16.1.2	TCP	ftp-data > 1074 [ACK] Seq=1 Ack=1 win=5840 Len=0 TSV=12998375 TSER=0
49	26.487052	192.168.254.254	172.16.1.2	FTP	Response: 150 Here comes the directory listing.
50	26.487252	192.168.254.254	172.16.1.2	FTP-DA	FTP Data: 61 bytes
51	26.487267	192.168.254.254	172.16.1.2	FTP	Response: 226 Directory send OK.

Figure 8. Wireshark Capture of an FTP Session with a Web Browser

3. Move into the middle Wireshark window and expand the FTP protocol. FTP communicates using codes, similar to HTTP.

What is the FTP server response 220?

When the FTP server issued a Response: 331 Please specify the password, what was the web browser reply?

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

When data is transferred or with simple directory listings, a new port is opened. This is called the transfer mode. The transfer mode can be either active or passive. In active mode, the server opens a TCP session to the FTP client and transfers data across that port. The FTP server source port number is 20, and the FTP client port number is some number above 1023. In passive mode, however, the client opens a new port to the server for data transfer. Both port numbers are above 1023.

What is the FTP-DATA port number used by the FTP server?

4. Open the Wireshark capture FTP_Web_Browser_Client, and observe the FTP communication. Although the clients are different, the commands are similar.

Step 5: 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. See Figure 9.

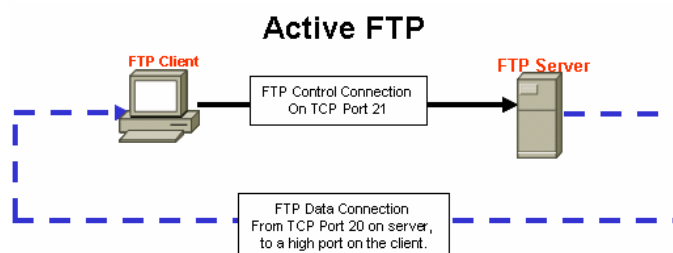


Figure 9.

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, the 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. See Figure 10.

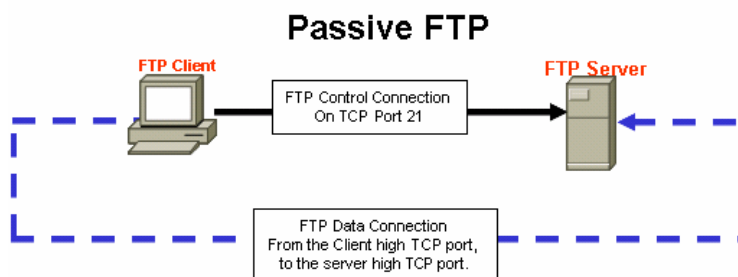


Figure 10.

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.

Task 4: Reflection

Both HTTP and FTP protocols rely on TCP to communicate. TCP manages the connection between client and server to ensure datagram delivery.

A client application may be either a web browser or command line utility, but each must send and receive messages that can be correctly interpreted. The communication protocol is normally defined in an RFC.

The FTP client must authenticate to the FTP server, even if the authentication is open to the world. User Anonymous normally has restricted access to the FTP server and cannot upload files.

An HTTP session begins when a request is made to the HTTP server and ends when the response has been acknowledged by the HTTP client. An FTP session, however, lasts until the client signals that it is leaving with the **quit** command.

HTTP uses a single protocol to communicate with the HTTP server. The server listens on port 80 for client connections. FTP, however, uses two protocols. The FTP server listens on TCP port 21, as the command line. Depending on the transfer mode, the server or client may initiate the data connection.

Multiple Application layer protocols can be accessed through a simple web browser. While only HTTP and FTP were examined, Telnet and Gopher may also be supported on the browser. The browser acts as a client to the server, sending requests and processing replies.

Task 5: Challenge

Enabling Wireshark capture, use a web browser to browse to R2 at <http://172.16.255.254/level/7/exec> or use a Telnet client to connect to a Cisco device such as S1-Central or R2-Central. Observe the HTTP or Telnet protocol behavior. Issue some commands to observe the results.

How is the Application layer protocol Telnet similar to HTTP and FTP? How is TELNET different?

Task 6: Clean Up

If Wireshark was installed on the pod host computer for this lab, the instructor may want the application removed. To remove Wireshark, click **Start > Control Panel > Add or Remove Programs**. Scroll to the bottom of the list, right-click on **Wireshark**, and click **Remove**.

If downloaded files need to be removed from the host pod computer, delete all files retrieved from the FTP server.

Unless directed otherwise by the instructor, turn off power to the host computers. Remove anything that was brought into the lab, and leave the room ready for the next class.