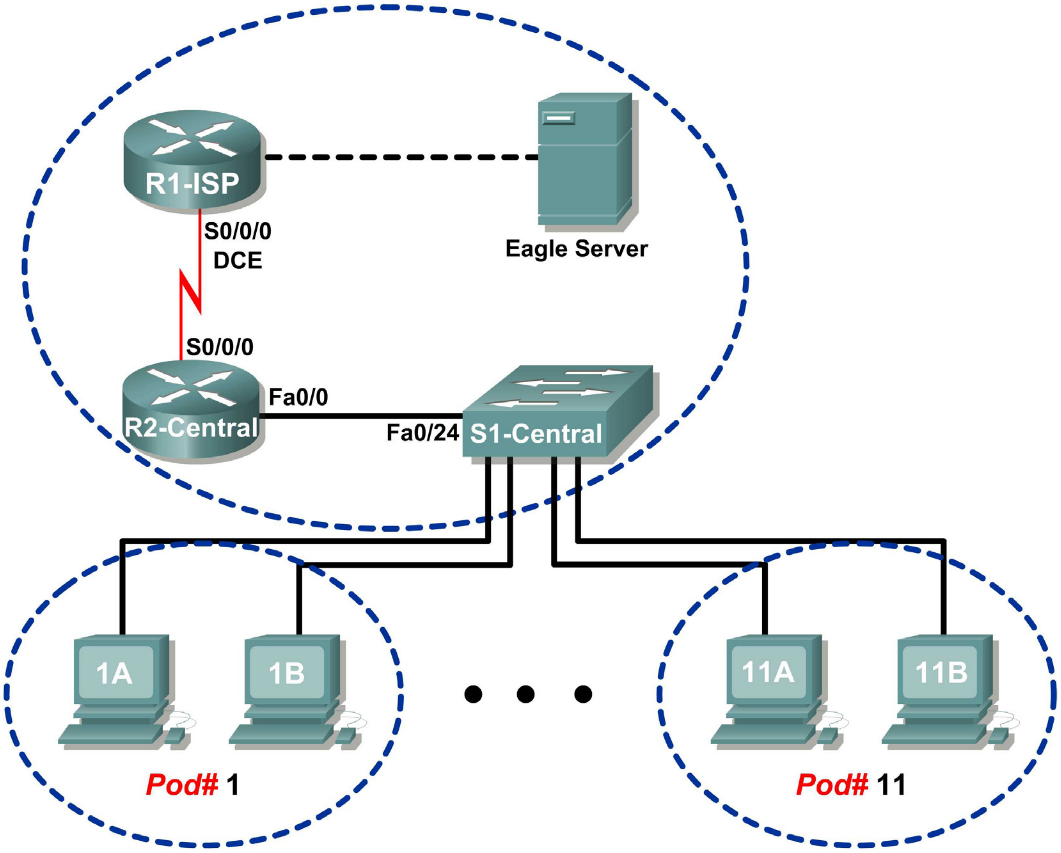
Lab 4.5.1: Observing TCP and UDP using Netstat

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
host*Pod#*A	N/A	172.16.*Pod#*.1	255.255.0.0	172.16.255.254
host*Pod#*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

- Explain common **netstat** command parameters and outputs.
- Use **netstat** to examine protocol information on a pod host computer.

Background

netstat is an abbreviation for the network statistics utility, available on both Windows and Unix / Linux computers. Passing optional parameters with the command will change output information. **netstat** displays incoming and outgoing network connections (TCP and UDP), host computer routing table information, and interface statistics.

Scenario

In this lab the student will examine the **netstat** command on a pod host computer, and adjust **netstat** output options to analyze and understand TCP/IP Transport Layer protocol status.

Task 1: Explain common netstat command parameters and outputs.

Open a terminal window by clicking on Start | Run. Type **cmd**, and press **OK**.

To display help information about the **netstat** command, use the **/?** options, as shown:

```
C:\> netstat /? <ENTER>
```

Use the output of the **netstat** /? command as reference to fill in the appropriate option that best matches the description:

Option	Description
-a	Display all connections and listening ports.
-n	Display addresses and port numbers in numerical form.
5	Redisplay statistics every five seconds. Press CTRL+C to stop redisplaying statistics.
-p	Shows connections for the protocol specified by proto; proto may be any of: TCP, UDP, TCPv6, or UDPv6. If used with the -s option to display per-protocol statistics, proto may be any of: IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, or UDPv6.
-an 5	Redisplay all connections and listening ports every 30 seconds.
no options	Display only open connections. This is a tricky problem.

When **netstat** statistics are displayed for TCP connections, the TCP state is displayed. During the life of a TCP connection, the connection passes through a series of states. The following table is a summary of TCP states, compiled from RFC 793, Transmission Control Protocol, September, 1981, as reported by **netstat**:

State	Connection Description
`LISTEN`	The local connection is waiting for a connection request from any remote device.
`ESTABLISHED`	The connection is open, and data may be exchanged through the connection. This is the normal state for the data transfer phase of the connection.
`TIME-WAIT`	The local connection is waiting a default period of time after sending a connection termination request before closing the connection. This is a normal condition, and will normally last between 30 - 120 seconds.
`CLOSE-WAIT`	The connection is closed, but is waiting for a termination request from the local user.
`SYN-SENT`	The local connection is waiting for a response after sending a connection request. The connection should transition quickly through this state.
`SYN_RECEIVED`	The local connection is waiting for a confirming connection request acknowledgment. The connection should transition quickly through this state. Multiple connections in SYN_RECEIVED state may indicate a TCP SYN attack.

IP addresses displayed by **`netstat`** fall into several categories:

IP Address	Description
`127.0.0.1`	This address refers to the local host, or this computer.
`0.0.0.0`	A global address, meaning “`ANY`”.
`Remote Address`	The address of the remote device that has a connection with this computer.

Task 2: Use `netstat` to Examine Protocol Information on a Pod Host Computer.

Step 1: Use `netstat` to view existing connections.

From the terminal window in Task 1, above, issue the command **`netstat –a`**:

```
C:\> netstat –a <ENTER>
```

A table will be displayed that lists protocol (TCP and UDP), Local address, Foreign address, and State information. Addresses and protocols that can be translated into names are displayed.

The `–n` option forces netstat to display output in raw format. From the terminal window, issue the command **`netstat –an`**:

```
C:\> netstat –an <ENTER>
```

Use the window vertical scroll bar to go back and forth between the outputs of the two commands. Compare outputs, noting how well-known port numbers are changed to names.

Write down three TCP and three UDP connections from the **netstat -a** output, and the corresponding translated port numbers from the **netstat -an** output. If there are fewer than three connections that translate, note that in your table.

Connection	Proto Local Address Foreign Address State
	TCP GW-desktop-hom:epmap GW-desktop-hom:0 LISTENING
	TCP 0.0.0.0:135 0.0.0.0:0 LISTENING
	TCP GW-desktop-hom:microsoft-ds GW-desktop-hom:0 LISTENING
	TCP 0.0.0.0:445 0.0.0.0:0 LISTENING
	TCP GW-desktop-hom:netbios-ssn GW-desktop-hom:0 LISTENING
	TCP 192.168.254.1:139 0.0.0.0:0 LISTENING
	UDP GW-desktop-hom:ntp *:*
	UDP 192.168.254.1:123 *:*
	UDP GW-desktop-hom:netbios-ns *:*
	UDP 192.168.254.1:137 *:*
	UDP GW-desktop-hom:netbios-dgm *:*
	UDP 192.168.254.1:138 *:*

Refer to the following **netstat** output. A new network engineer suspects that his host computer has been compromised by an outside attack against ports 1070 and 1071. How would you respond?

```
C:\> netstat -n
Active Connections
Proto  Local Address          Foreign Address        State
TCP    127.0.0.1:1070         127.0.0.1:1071         ESTABLISHED
TCP    127.0.0.1:1071         127.0.0.1:1070         ESTABLISHED
C:\>
```

The host computer has connected to itself

Step 2: Establish multiple concurrent TCP connections and record netstat output.

In this task, several simultaneous connections will be made with Eagle Server. The venerable **telnet** command will be used to access Eagle Server network services, thus providing several protocols to examine with netstat.

Open an additional four terminal windows. Arrange the windows so that all are visible. The four terminal windows that will be used for telnet connections to Eagle Server can be relatively small, approximately ½ screen width by ¼ screen height. The terminal windows that will be used to collect connection information should be ½ screen width by full screen height.

Several network services on Eagle Server will respond to a telnet connection. We will use:

- DNS- domain name server, port 53
- FTP- FTP server, port 21
- SMTP- SMTP mail server, port 25
- TELNET- Telnet server, port 23

Why should telnet to UDP ports fail?

____

Telnet is a TCP protocol

To close a telnet connection, press the `<CTRL> ]` keys together. That will bring up the telnet prompt, `Microsoft Telnet>`. Type **`quit <ENTER>`** to close the session.

In the first telnet terminal window, telnet to Eagle Server on port 53. In the second terminal window, telnet on port 21. In the third terminal window, telnet on port 25. In the fourth terminal window, telnet on port 23. The command for a telnet connection on port 21 is shown below:

```
C:\> telnet eagle-server.example.com 53
```

In the large terminal window, record established connections with Eagle Server. Output should look similar to the following. If typing is slow, a connection may close before all connections have been made. Eventually, connections should terminate from inactivity.

```
Proto  Local Address          Foreign Address        State
TCP    192.168.254.1:1688     192.168.254.254:21     ESTABLISHED
TCP    192.168.254.1:1691     192.168.254.254:25     ESTABLISHED
TCP    192.168.254.1:1693     192.168.254.254:53     ESTABLISHED
TCP    192.168.254.1:1694     192.168.254.254:23     ESTABLISHED
```

Task 3: Reflection.

The **`netstat`** utility displays incoming and outgoing network connections (TCP and UDP), host computer routing table information, and interface statistics.

Task 4: Challenge.

Close Established sessions abruptly (close the terminal window), and issue the **`netstat -an`** command. Try to view connections in stages different from ESTABLISHED.

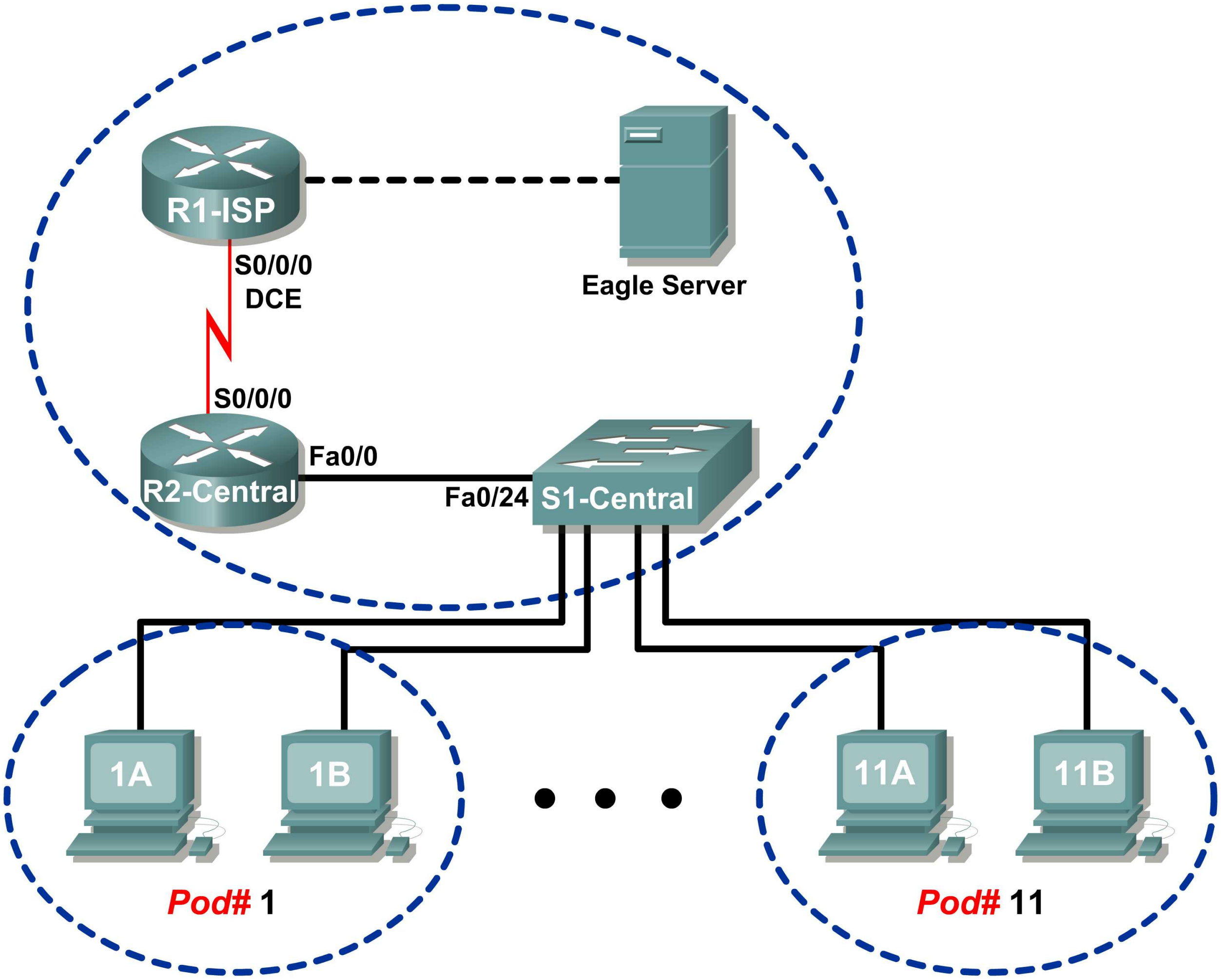
Task 5: Cleanup.

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.

Cisco | Networking Academy®
Mind Wide Open™

Lab 4.5.2: TCP/IP Transport Layer Protocols, TCP and UDP

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
host*Pod#*A	N/A	172.16.*Pod#*.1	255.255.0.0	172.16.255.254
host*Pod#*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

- Identify TCP header fields and operation using a Wireshark FTP session capture.
- Identify UDP header fields and operation using a Wireshark TFTP session capture.

Background

The two protocols in the TCP/IP Transport Layer are the transmission control protocol (TCP), defined in RFC 761, January, 1980, and user datagram protocol (UDP), defined in RFC 768, August, 1980. Both protocols support upper-layer protocol communication. For example, TCP is used to provide Transport Layer support for the HTTP and FTP protocols, among others. UDP provides Transport Layer support for domain name services (DNS) and trivial file transfer protocol (TFTP), among others.

The ability to understand the parts of the TCP and UDP headers and operation are a critical skill for network engineers.

Scenario

Using Wireshark capture, analyze TCP and UDP protocol header fields for file transfers between the host computer and Eagle Server. If Wireshark has not been loaded on the host pod computer, it may be downloaded from URL `ftp://eagle-server.example.com/pub/eagle_labs/eagle1/chapter4/`, file `wireshark-setup-0.99.4.exe`.

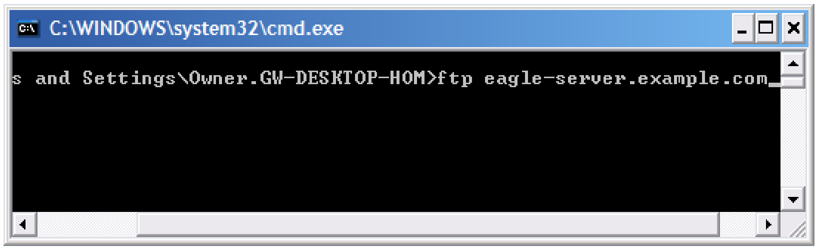
Windows command line utilities `ftp` and `tftp` will be used to connect to Eagle Server and download files.

Task 1: Identify TCP Header Fields and Operation using a Wireshark FTP Session Capture.

Step 1: Capture a FTP session.

TCP sessions are well controlled and managed by information exchanged in the TCP header fields. In this task, a FTP session will be made to Eagle Server. When finished, the session capture will be analyzed. Windows computers use the FTP client, **`ftp`**, to connect to the FTP server. A command line window will start the FTP session, and the text configuration file for S1-central from Eagle Server will be downloaded, `/pub/eagle_labs/eagle1/chapter4/s1-central`, to the host computer.

Open a command line window by clicking on Start | Run, type cmd, then press OK.



```
s and Settings\Owner.GW-DESKTOP-HOM>ftp eagle-server.example.com_
```

Figure 1. Command line window.

A window similar to Figure 1 should open.

Start a Wireshark capture on the interface that has IP address `172.16.`*`Pod#`*`.[1-2]`.

Start an FTP connection to Eagle Server. Type the command:

```
> ftp eagle-server.example.com
```

When prompted for a user id, type **`anonymous`**. When prompted for a password, press **`<ENTER>`**.

Change the FTP directory to `/pub/eagle_labs/eagle1/chapter4/`:

```
ftp> cd /pub/eagle_labs/eagle1/chapter4/
```

Download the file `s1-central`:

```
ftp> get s1-central
```

When finished, terminate the FTP sessions in each command line window with the FTP **`quit`** command:

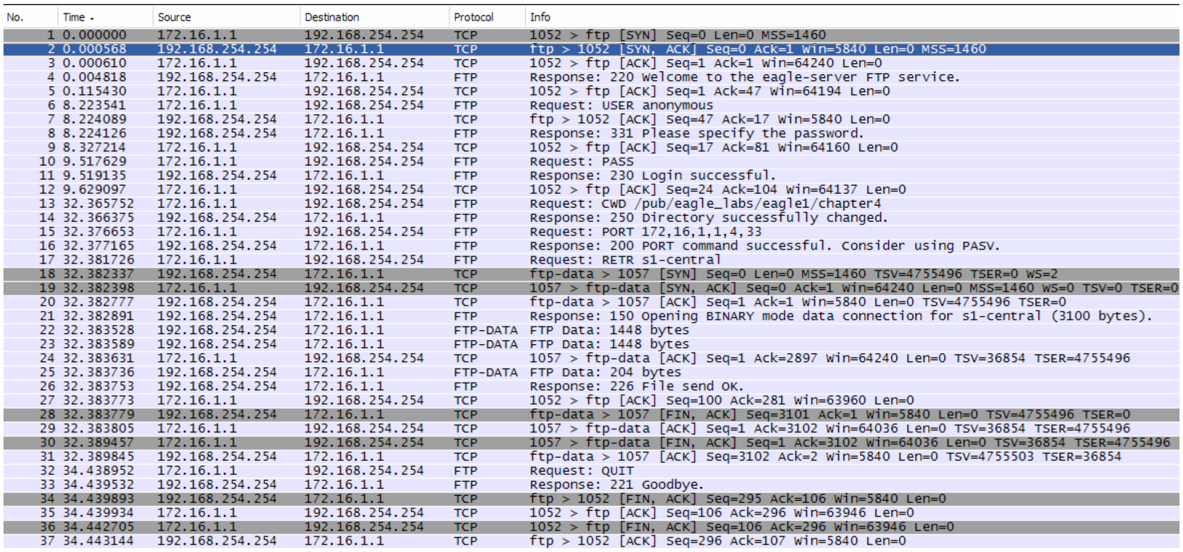
```
ftp> quit
```

Close the command line window with the command **`exit`**:

```
> exit
```

Stop the Wireshark capture.

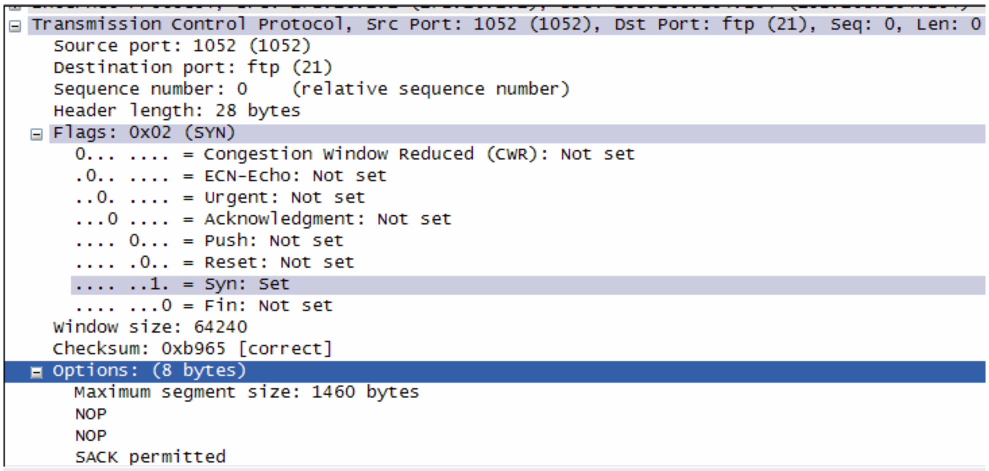
Step 2: Analyze the TCP fields.



```
No.  Time -     Source           Destination      Protocol  Info
  1 0.000000    172.16.1.1       192.168.254.254  TCP       1052 > ftp [SYN] Seq=0 Len=0 MSS=1460
  2 0.000568    192.168.254.254  172.16.1.1       TCP       ftp > 1052 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
  3 0.000610    172.16.1.1       192.168.254.254  TCP       1052 > ftp [ACK] Seq=1 Ack=1 Win=64240 Len=0
  4 0.004818    192.168.254.254  172.16.1.1       FTP       Response: 220 Welcome to the eagle-server FTP service.
  5 0.115430    172.16.1.1       192.168.254.254  TCP       1052 > ftp [ACK] Seq=1 Ack=47 Win=64194 Len=0
  6 8.223541    172.16.1.1       192.168.254.254  FTP       Request: USER anonymous
  7 8.224089    192.168.254.254  172.16.1.1       TCP       ftp > 1052 [ACK] Seq=47 Ack=17 Win=5840 Len=0
  8 8.224126    192.168.254.254  172.16.1.1       FTP       Response: 331 Please specify the password.
  9 8.327214    172.16.1.1       192.168.254.254  TCP       1052 > ftp [ACK] Seq=17 Ack=81 Win=64160 Len=0
 10 9.517629    172.16.1.1       192.168.254.254  FTP       Request: PASS
 11 9.519135    192.168.254.254  172.16.1.1       FTP       Response: 230 Login successful.
 12 9.629097    172.16.1.1       192.168.254.254  TCP       1052 > ftp [ACK] Seq=24 Ack=104 Win=64137 Len=0
 13 32.365752   172.16.1.1       192.168.254.254  FTP       Request: CWD /pub/eagle_labs/eagle1/chapter4
 14 32.366375   192.168.254.254  172.16.1.1       FTP       Response: 250 Directory successfully changed.
 15 32.376653   172.16.1.1       192.168.254.254  FTP       Request: PORT 172,16,1,1,4,33
 16 32.377165   192.168.254.254  172.16.1.1       FTP       Response: 200 PORT command successful. Consider using PASV.
 17 32.381726   172.16.1.1       192.168.254.254  FTP       Request: RETR s1-central
 18 32.382337   192.168.254.254  172.16.1.1       TCP       ftp-data > 1057 [SYN] Seq=0 Len=0 MSS=1460 TSV=4755496 TSER=0 WS=2
 19 32.382398   172.16.1.1       192.168.254.254  TCP       1057 > ftp-data [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
 20 32.382777   192.168.254.254  172.16.1.1       TCP       ftp-data > 1057 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=4755496 TSER=0
 21 32.382891   192.168.254.254  172.16.1.1       FTP       Response: 150 Opening BINARY mode data connection for s1-central (3100 bytes).
 22 32.383528   192.168.254.254  172.16.1.1       FTP-DATA  FTP Data: 1448 bytes
 23 32.383589   192.168.254.254  172.16.1.1       FTP-DATA  FTP Data: 1448 bytes
 24 32.383631   172.16.1.1       192.168.254.254  TCP       1057 > ftp-data [ACK] Seq=1 Ack=2897 Win=64240 Len=0 TSV=36854 TSER=4755496
 25 32.383736   192.168.254.254  172.16.1.1       FTP-DATA  FTP Data: 204 bytes
 26 32.383753   192.168.254.254  172.16.1.1       FTP       Response: 226 File send OK.
 27 32.383773   172.16.1.1       192.168.254.254  TCP       1052 > ftp [ACK] Seq=100 Ack=281 Win=63960 Len=0
 28 32.383779   192.168.254.254  172.16.1.1       TCP       ftp-data > 1057 [FIN, ACK] Seq=3101 Ack=1 Win=5840 Len=0 TSV=4755496 TSER=0
 29 32.383805   172.16.1.1       192.168.254.254  TCP       1057 > ftp-data [ACK] Seq=1 Ack=3102 Win=64036 Len=0 TSV=36854 TSER=4755496
 30 32.389457   172.16.1.1       192.168.254.254  TCP       1057 > ftp-data [FIN, ACK] Seq=1 Ack=3102 Win=64036 Len=0 TSV=36854 TSER=4755496
 31 32.389845   192.168.254.254  172.16.1.1       TCP       ftp-data > 1057 [ACK] Seq=3102 Ack=2 Win=5840 Len=0 TSV=4755503 TSER=36854
 32 34.438952   172.16.1.1       192.168.254.254  FTP       Request: QUIT
 33 34.439532   192.168.254.254  172.16.1.1       FTP       Response: 221 Goodbye.
 34 34.439893   192.168.254.254  172.16.1.1       TCP       ftp > 1052 [FIN, ACK] Seq=295 Ack=106 Win=5840 Len=0
 35 34.439934   172.16.1.1       192.168.254.254  TCP       1052 > ftp [ACK] Seq=106 Ack=296 Win=63946 Len=0
 36 34.442705   172.16.1.1       192.168.254.254  TCP       1052 > ftp [FIN, ACK] Seq=106 Ack=296 Win=63946 Len=0
 37 34.443144   192.168.254.254  172.16.1.1       TCP       ftp > 1052 [ACK] Seq=296 Ack=107 Win=5840 Len=0
```

Figure 2. FTP capture.

Switch to the Wireshark capture windows. The top window contains summary information for each captured record. Student capture should be similar to the capture shown in Figure 2. Before delving into TCP packet details, an explanation of the summary information is needed. When the FTP client is connected to the FTP server, the Transport Layer protocol TCP created a reliable session. TCP is routinely used during a session to control datagram delivery, verify datagram arrival, and manage window size. For each exchange of data between the FTP client and FTP server, a new TCP session is started. At the conclusion of the data transfer, the TCP session is closed. Finally, when the FTP session is finished TCP performs an orderly shutdown and termination.



```
⊟ Transmission Control Protocol, Src Port: 1052 (1052), Dst Port: ftp (21), Seq: 0, Len: 0
    Source port: 1052 (1052)
    Destination port: ftp (21)
    Sequence number: 0    (relative sequence number)
    Header length: 28 bytes
  ⊟ Flags: 0x02 (SYN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgment: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 64240
    Checksum: 0xb965 [correct]
  ⊟ Options: (8 bytes)
      Maximum segment size: 1460 bytes
      NOP
      NOP
      SACK permitted
```

Figure 3. Wireshark capture of a TCP datagram.

In Wireshark, detailed TCP information is available in the middle window. Highlight the first TCP datagram from the host computer, and move the mouse pointer to the middle window. It may be necessary to adjust the middle window and expand the TCP record by clicking on the protocol expand box. The expanded TCP datagram should look similar to Figure 3.

How is the first datagram in a TCP session identified?

Originates from the host computer on a port higher than 1023,

to the Eagle Server 192.168.254.254 on port 21

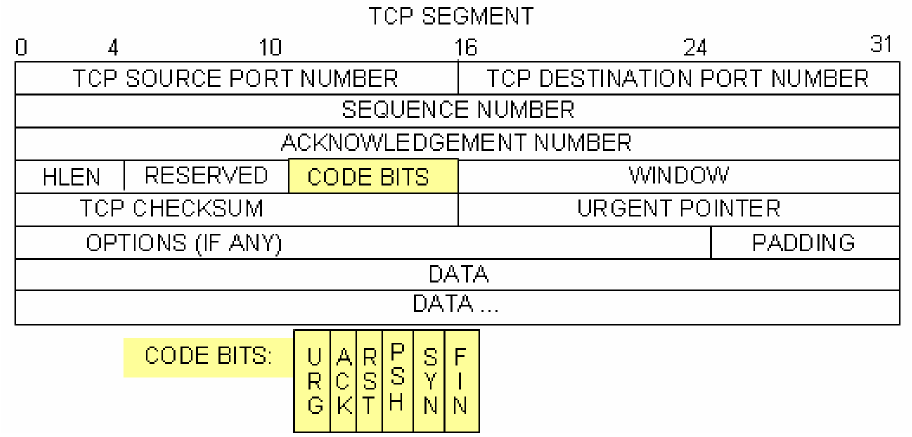


Figure 4. TCP packet fields.

Refer to Figure 4, a TCP datagram diagram. An explanation of each field is provided to refresh the student's memory:

- **TCP Source port number** belongs to the TCP session host that opened a connection. The value is normally a random value above 1023.
- **Destination port number** is used to identify the upper layer protocol or application on the remote site. The values in the range 0–1023 represent the so called "well known ports" and are associated with popular services and applications (as described in RFC 1700, such as telnet, File Transfer Protocol (FTP), HyperText Transfer Protocol (HTTP), etc). The quadruple field combination (Source IP Address, Source Port, Destination IP Address, Destination Port) uniquely identifies the session to both sender and receiver.
- **Sequence number** specifies the number of the last octet in a segment.
- **Acknowledgment number** specifies the next octet expected by the receiver.
- **Code Bits** have a special meaning in session management and in the treatment of segments. Among interesting values are:
  - ACK (Acknowledgement of a segment receipt),
  - SYN (Synchronize, only set when a new TCP session is negotiated during the TCP three-way handshake).
  - FIN (Finish, request to close the TCP session).
- **Window size** is the value of the sliding window - how many octets can be sent before waiting for an acknowledgement.
- **Urgent pointer** is only used with an URG (Urgent) flag - when the sender needs to send urgent data to the receiver.
- **Options**: The only option currently defined is the maximum TCP segment size (optional value).

Using the Wireshark capture of the first TCP session start-up (SYN bit set to 1), fill in information about the TCP header:

From pod host computer to Eagle Server (only the SYN bit is set to 1):

Source IP Address: 172.16.___.___	172.16.1.1
Destination IP Address: ________________	192.168.254.254
Source port number: ________________	1098
Destination port number: ________________	21
Sequence number: ________________	0
Acknowledgement number: ____________	28 bytes
Header length: ________________	
Window size: ________________	62440

From Eagle Server to pod host computer (only SYN and ACK bits are set to 1):

Source IP Address: ________________	192.168.254.254
Destination IP Address: 172.16.___.___	172.16.1.1
Source port number: ________________	21
Destination port number: ________________	1098
Sequence number: ________________	0
Acknowledgement number: ____________	1
Header length: ________________	28 bytes
Window size: ________________	5730

From pod host computer to Eagle Server (only ACK bit is set to 1):

Source IP Address: 172.16.___.___	172.16.1.1
Destination IP Address: ________________	192.168.254.254
Source port number: ________________	1098
Destination port number: ________________	21
Sequence number: ________________	1
Acknowledgement number: ____________	1
Header length: ________________	20 bytes
Window size: ________________	62440

Ignoring the TCP session started when a data transfer occurred, how many other TCP datagrams contained a SYN bit?

SYN is only used at the beginning

Attackers take advantage of the three-way handshake by initiating a “half-open” connection. In this sequence, the opening TCP session sends a TCP datagram with the SYN bit set and the receiver sends a related TCP datagram with the SYN ACK bits set. A final ACK bit is never sent to finish the TCP handshake. Instead, a new TCP connection is started in half-open fashion. With sufficient TCP sessions in the half-open state, the receiving computer may exhaust resources and crash. A crash could involve a loss of networking services, or corrupt the operating system. In either case the attacker has won, networking service has been stopped on the receiver. This is one example of a denial-of-service (DoS) attack.

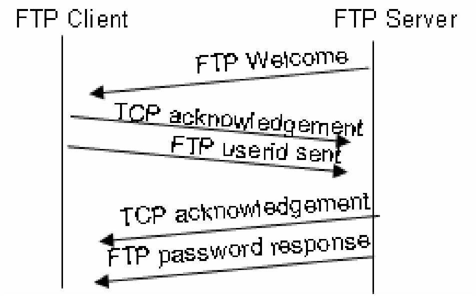


Figure 5. TCP session management.

The FTP client and server communicate between each other, unaware and uncaring that TCP has control and management over the session. When the FTP server sends a Response: 220 to the FTP client, the TCP session on the FTP client sends an acknowledgment to the TCP session on Eagle Server. This sequence is shown in Figure 5, and is visible in the Wireshark capture.

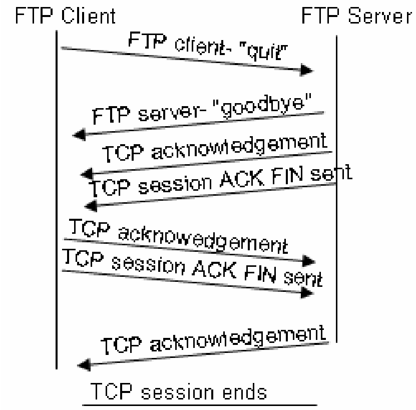


Figure 6. Orderly TCP session termination.

When the FTP session has finished, the FTP client sends a command to "`quit`". The FTP server acknowledges the FTP termination with a `Response :221 Goodbye`. At this time the FTP server TCP session sends a TCP datagram to the FTP client, announcing the termination of the TCP session. The FTP client TCP session acknowledges receipt of the termination datagram, then sends its own TCP session termination. When the originator of the TCP termination, FTP server, receives a duplicate termination, an ACK datagram is sent to acknowledge the termination and the TCP session is closed. This sequence is shown in Figure 6, and visible in the Wireshark capture.

Without an orderly termination, such as when the connection is broken, the TCP sessions will wait a certain period of time until closing. The default timeout value varies, but is normally 5 minutes.

Task 2: Identify UDP header fields and operation using a Wireshark TFTP session capture.

Step 1: Capture a TFTP session.

Following the procedure in Task 1 above, open a command line window. The TFTP command has a different syntax than FTP. For example, there is no authentication. Also, there are only two commands, **`get`**, to retrieve a file, and **`put`**, to send a file.

```
>tftp -help

Transfers files to and from a remote computer running the TFTP service.

TFTP [-i] host [GET | PUT] source [destination]

        -i       Specifies binary image transfer mode (also called
                 octet). In binary image mode the file is moved
                 literally, byte by byte. Use this mode when
                 transferring binary files.
        host     Specifies the local or remote host.
        GET      Transfers the file destination on the remote host to
                 the file source on the local host.
        PUT      Transfers the file source on the local host to
                 the file destination on the remote host.
        source   Specifies the file to transfer.
  destination    Specifies where to transfer the file.
```

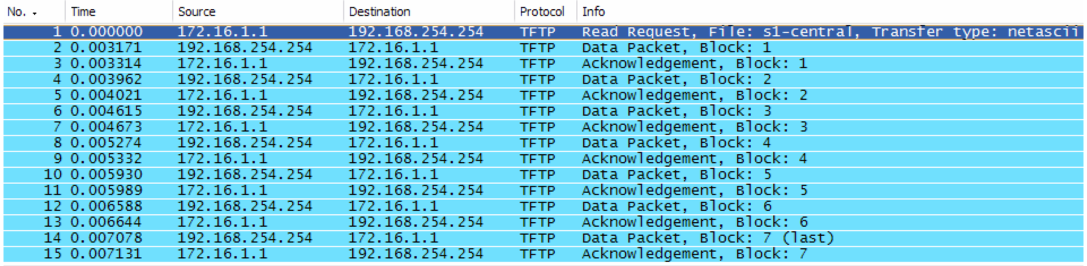
Table 1. TFTP syntax for a Windows TFTP client.

Table 1 contains Windows TFTP client syntax. The TFTP server has it's own directory on Eagle Server, `/tftpboot`, which is different from the directory structure supported by the FTP server. No authentication is supported.

Start a Wireshark capture, then download the `s1-central` configuration file from Eagle Server with the Windows TFTP client. The command and syntax to perform this is shown below:

```
>tftp eagle-server.example.com get s1-central
```

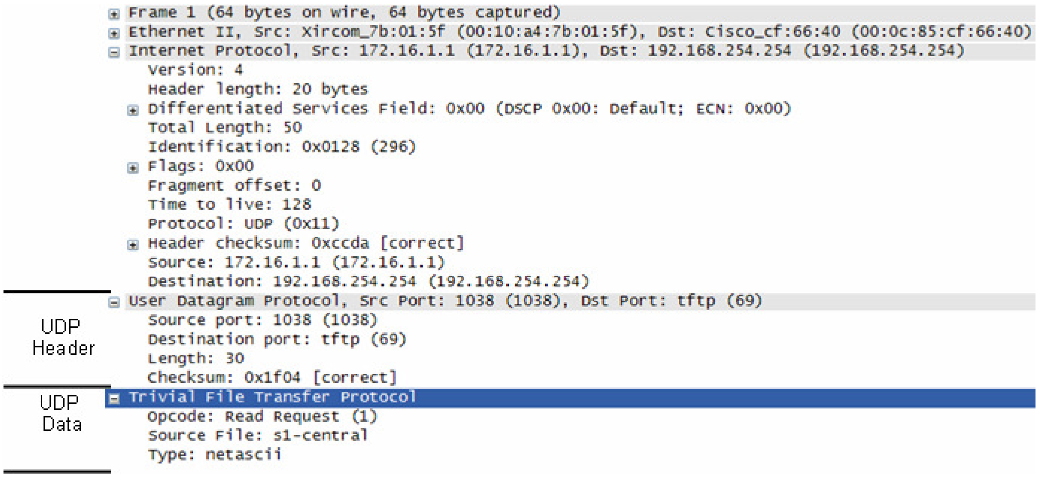
Step 2: Analyze the UDP fields.



No.	Time	Source	Destination	Protocol	Info
1	0.000000	172.16.1.1	192.168.254.254	TFTP	Read Request, File: s1-central, Transfer type: netascii
2	0.003171	192.168.254.254	172.16.1.1	TFTP	Data Packet, Block: 1
3	0.003314	172.16.1.1	192.168.254.254	TFTP	Acknowledgement, Block: 1
4	0.003962	192.168.254.254	172.16.1.1	TFTP	Data Packet, Block: 2
5	0.004021	172.16.1.1	192.168.254.254	TFTP	Acknowledgement, Block: 2
6	0.004615	192.168.254.254	172.16.1.1	TFTP	Data Packet, Block: 3
7	0.004673	172.16.1.1	192.168.254.254	TFTP	Acknowledgement, Block: 3
8	0.005274	192.168.254.254	172.16.1.1	TFTP	Data Packet, Block: 4
9	0.005332	172.16.1.1	192.168.254.254	TFTP	Acknowledgement, Block: 4
10	0.005930	192.168.254.254	172.16.1.1	TFTP	Data Packet, Block: 5
11	0.005989	172.16.1.1	192.168.254.254	TFTP	Acknowledgement, Block: 5
12	0.006588	192.168.254.254	172.16.1.1	TFTP	Data Packet, Block: 6
13	0.006644	172.16.1.1	192.168.254.254	TFTP	Acknowledgement, Block: 6
14	0.007078	192.168.254.254	172.16.1.1	TFTP	Data Packet, Block: 7 (last)
15	0.007131	172.16.1.1	192.168.254.254	TFTP	Acknowledgement, Block: 7

Figure 7. Summary capture of a UDP session.

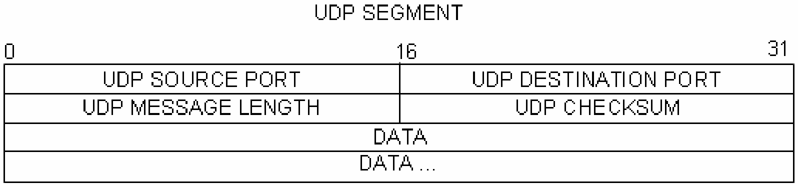
Switch to the Wireshark capture windows. Student capture should be similar to the capture shown in Figure 7. A TFTP transfer will be used to analyze Transport Layer UDP operation.



```
⊞ Frame 1 (64 bytes on wire, 64 bytes captured)
⊞ Ethernet II, Src: Xircom_7b:01:5f (00:10:a4:7b:01:5f), Dst: Cisco_cf:66:40 (00:0c:85:cf:66:40)
⊟ Internet Protocol, Src: 172.16.1.1 (172.16.1.1), Dst: 192.168.254.254 (192.168.254.254)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 50
    Identification: 0x0128 (296)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (0x11)
  ⊞ Header checksum: 0xccda [correct]
    Source: 172.16.1.1 (172.16.1.1)
    Destination: 192.168.254.254 (192.168.254.254)
⊟ User Datagram Protocol, Src Port: 1038 (1038), Dst Port: tftp (69)
    Source port: 1038 (1038)
    Destination port: tftp (69)
    Length: 30
    Checksum: 0x1f04 [correct]
⊟ Trivial File Transfer Protocol
    Opcode: Read Request (1)
    Source File: s1-central
    Type: netascii
```

Figure 8. Wireshark capture of a UDP datagram.

In Wireshark, detailed UDP information is available in the middle window. Highlight the first UDP datagram from the host computer, and move the mouse pointer to the middle window. It may be necessary to adjust the middle window and expand the UDP record by clicking on the protocol expand box. The expanded UDP datagram should look similar to Figure 8.



UDP SOURCE PORT	UDP DESTINATION PORT
UDP MESSAGE LENGTH	UDP CHECKSUM
DATA	
DATA ...	

Figure 9. UDP format.

Refer to Figure 9, a UDP datagram diagram. Header information is sparse, compared to the TCP datagram. There are similarities, however. Each UDP datagram is identified by the UDP source port and UDP destination port.

Using the Wireshark capture of the first UDP datagram, fill in information about the UDP header. The checksum value is a hexadecimal (base 16) value, denoted by the preceding 0x code:

Source IP Address: 172.16.___.___	172.16.1.1
Destination IP Address: ________________	192.168.254.254
Source port number: ________________	1038
Destination port number: ________________	69
UDP message length: _______________	30 bytes
UDP checksum: _______________	0x1f04

How does UDP verify datagram integrity?

Checksum is sent in the UDP Datagram.

if the checksum matches , the datagram is complete.

Examine the first packet returned from Eagle Server. Fill in information about the UDP header:

Source IP Address:	192.168.254.254
Destination IP Address: 172.16.___.___	172.16.1.1
Source port number: _______________	32799
Destination port number: _______________	1038
UDP message length: _______________	524 bytes
UDP checksum: 0x _______________	0xc5cf

Notice that the return UDP datagram has a different UDP source port, but this source port is used for the remainder of the TFTP transfer. Since there is no reliable connection, only the original source port used to begin the TFTP session is used to maintain the TFTP transfer.

Task 5: Reflection.

This lab provided students with the opportunity to analyze TCP and UDP protocol operations from captured FTP and TFTP sessions. TCP manages communication much differently from UDP, but reliability and guaranteed delivery requires additional control over the communication channel. UDP has less overhead and control, and the upper-layer protocol must provide some type of acknowledgement control. Both protocols, however, transport data between clients and servers using Application Layer protocols and are appropriate for the upper-layer protocol each supports.

Task 6: Challenge.

Since neither FTP nor TFTP are secure protocols, all data transferred is sent in clear text. This includes any user ids, passwords, or clear text file contents. Analyzing the upper-layer FTP session will quickly identify the user id, password, and configuration file passwords. Upper-layer TFTP data examination is a bit more complicated, but the data field can be examined and configuration user id and password information extracted.

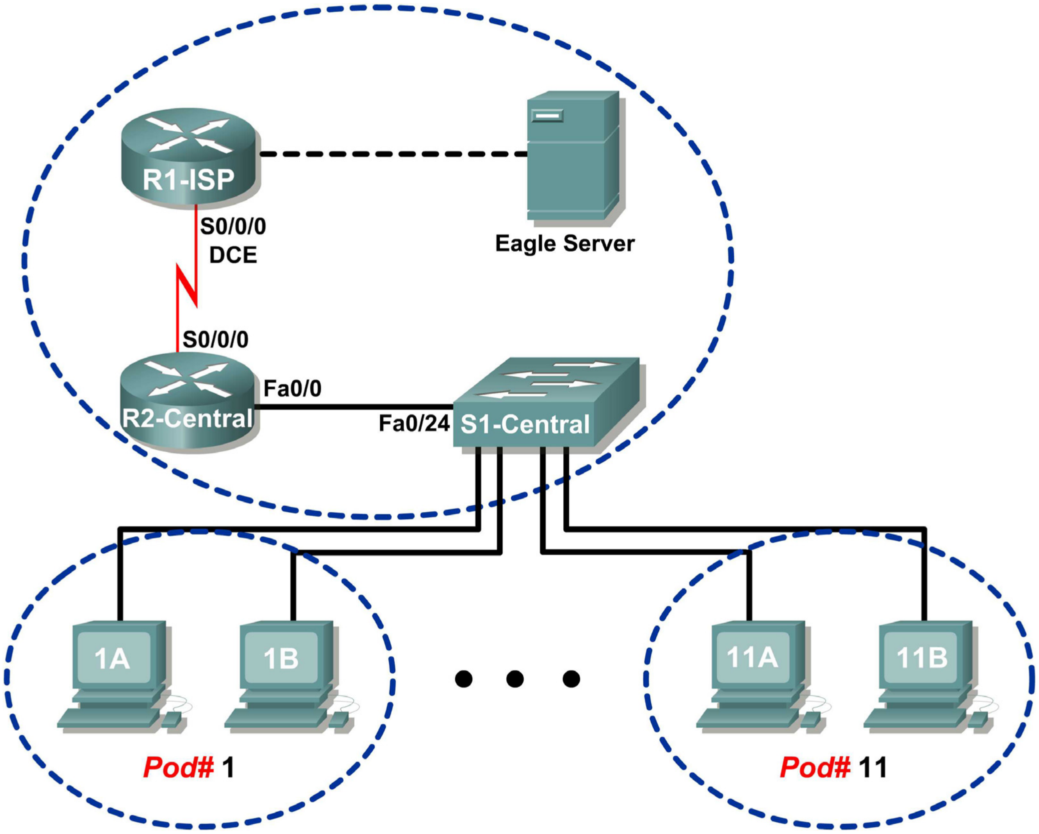
Task 7: Cleanup

During this lab several files were transferred to the host computer, and should be removed.

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.

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
host*Pod#*A	N/A	172.16.*Pod#*.1	255.255.0.0	172.16.255.254
host*Pod#*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.

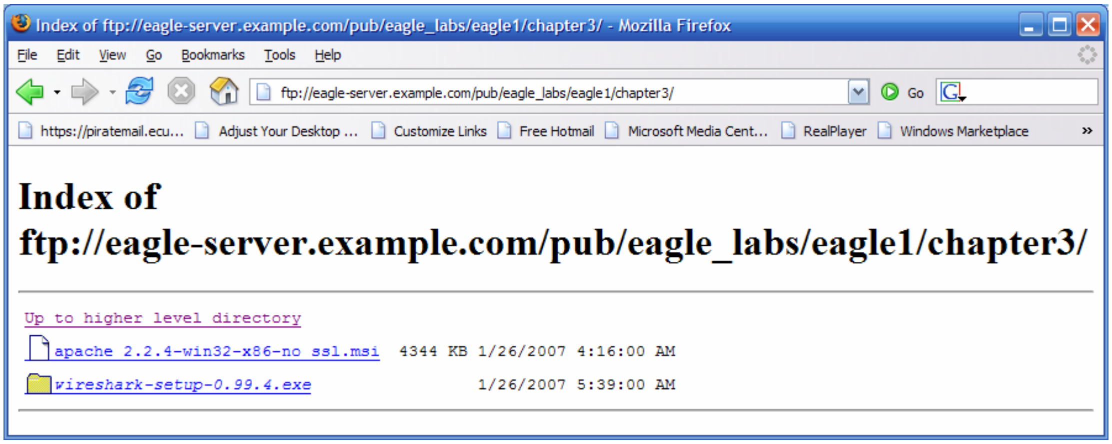
A web browser such as Internet Explorer or Firefox will be used to connect to the eagle-server network service. Eagle-server has several network services preconfigured, such as HTTP, waiting to respond to client requests.

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.


Index of ftp://eagle-server.example.com/pub/eagle_labs/eagle1/chapter3/ - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
ftp://eagle-server.example.com/pub/eagle_labs/eagle1/chapter3/
Go
https://piratemail.ecu... Adjust Your Desktop ... Customize Links Free Hotmail Microsoft Media Cent... RealPlayer Windows Marketplace

Index of ftp://eagle-server.example.com/pub/eagle_labs/eagle1/chapter3/

```
Up to higher level directory
apache_2.2.4-win32-x86-no_ssl.msi  4344 KB 1/26/2007 4:16:00 AM
wireshark-setup-0.99.4.exe                 1/26/2007 5:39:00 AM
```

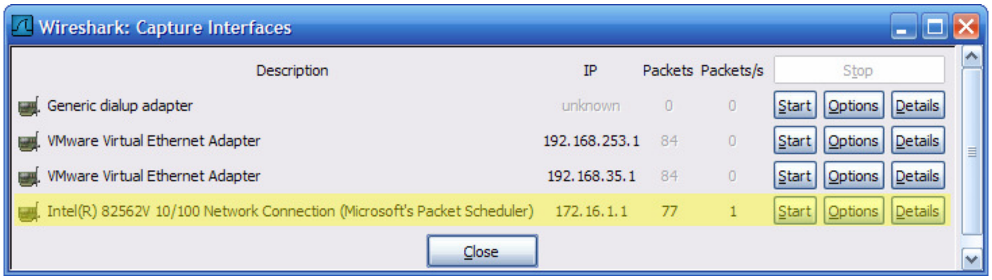
Figure 1. FTP Download for Wireshark

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.



Description	IP	Packets	Packets/s	
Generic dialup adapter	unknown	0	0	Start Options Details
VMware Virtual Ethernet Adapter	192.168.253.1	84	0	Start Options Details
VMware Virtual Ethernet Adapter	192.168.35.1	84	0	Start Options Details
Intel(R) 82562V 10/100 Network Connection (Microsoft's Packet Scheduler)	172.16.1.1	77	1	Start Options Details

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.

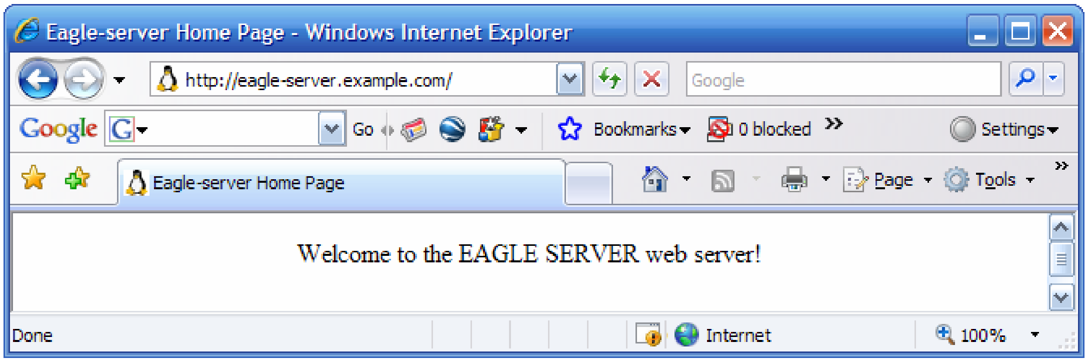
HTTP is an Application layer protocol, relying on lower level protocols such as TCP to establish and manage the communication channel. HTTP version 1.1 is defined in RFC 2616, dated 1999. This part of the lab will demonstrate how sessions between multiple web clients and the web server are kept separate.

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.



Welcome to the EAGLE SERVER web server!

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.

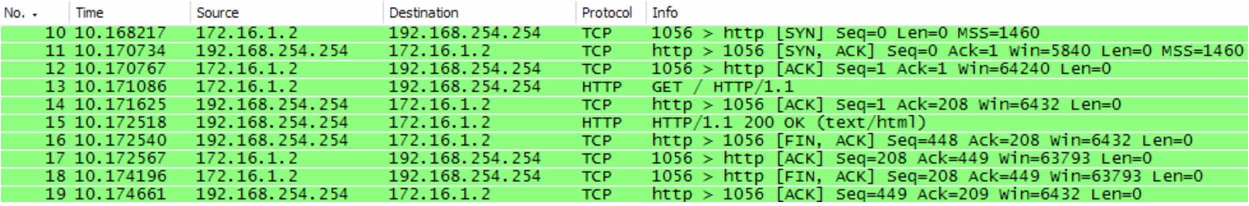
3. Open a second web browser, and connect to URL `http://eagle-server.example.com/page2.html`. This will display a different web page.

   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	172.16.1.2
Web server IP address	192.168.254.254
Transport layer protocol (UDP/TCP)	TCP
Web browser port number	1056
Web server port number	80

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

initiated by 172.16.1.2 to 192.168.254.254 with SYN flag

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

192.168.254.254 with FIN ACK

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?

HTTP

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	http/1.1
Request Method	get
* Request URI	\
Language	en-us

* 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?

200 ok from rfc2616

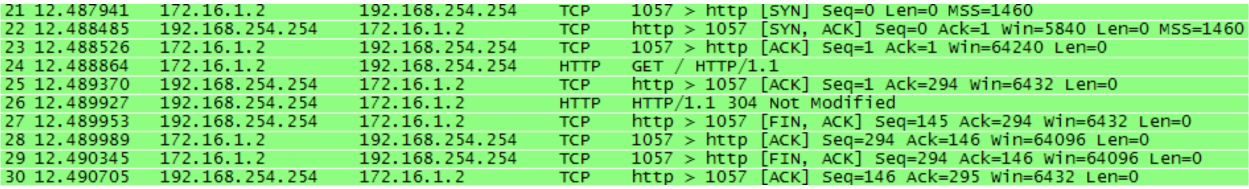
---

11. What does this response mean?

successful request

---

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)

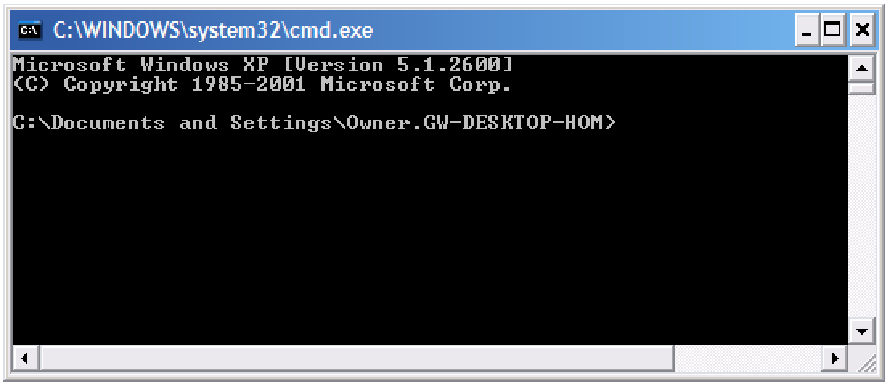
13. What is the ETAG response from the web server?

98072-b8-82da8740

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.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Owner.GW-DESKTOP-HOM>
```

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.

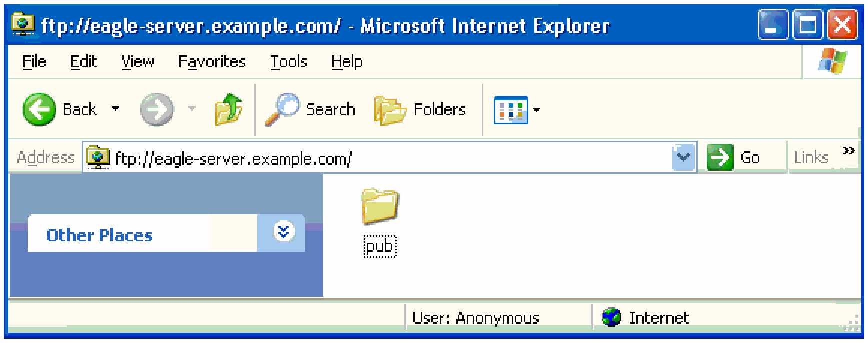
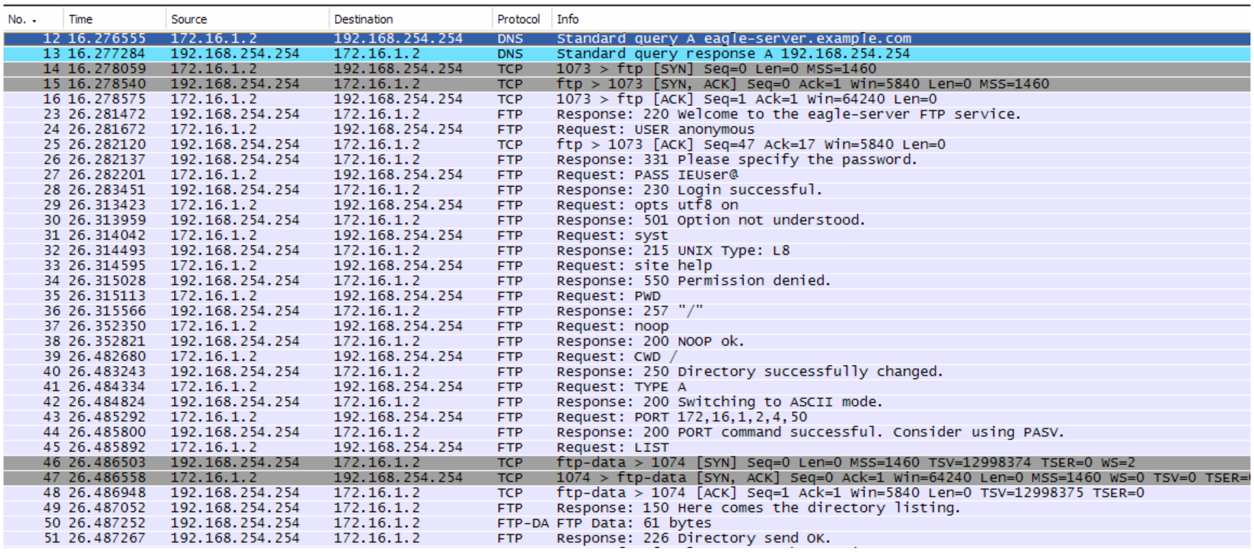

ftp://eagle-server.example.com/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Folders
Address ftp://eagle-server.example.com/
Go
Links
Other Places
pub
User: Anonymous
Internet

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. **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 IEUser@
   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?

ready for new request

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

IEUSER@

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

1073

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?
20

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.

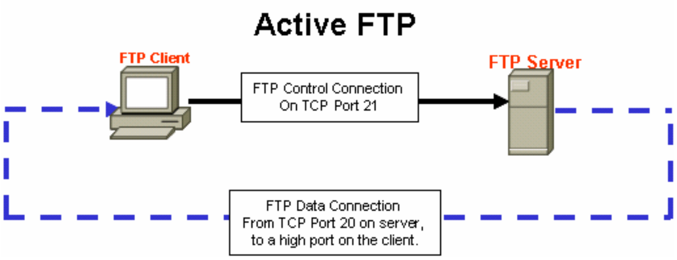

Active FTP
FTP Client
FTP Server
FTP Control Connection On TCP Port 21
FTP Data Connection From TCP Port 20 on server, to a high port on the client.

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.

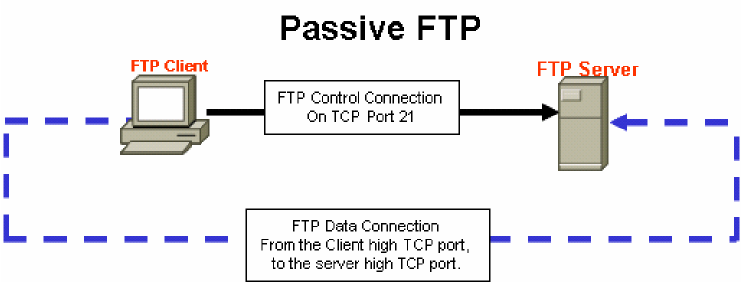

Passive FTP
FTP Client
FTP Server
FTP Control Connection On TCP Port 21
FTP Data Connection From the Client high TCP port, to the server high TCP port.

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 or command line Telnet client to connect to a Cisco device such as S1-Central or R2-Central. Observe the Telnet protocol behavior. Issue a **GET** request and observe the results.

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

they all use TCP protocol.

____

____

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.