TCP/IP Transport Layer Protocols, TCP and UDP

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 your host computer and FTP and TFTP servers.
Windows command line utilities `ftp` and `tftp` will be used to connect to the servers. If you do not have a tftp client enabled by default on your Windows OS you can enable it under Control Panel – Add Programs – Add Windows Components.
You can download a tftp server from http://www.jounin.net/tftpd32_download.html or use any similar simple tftp server.

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 any FTP server (e.g. ftp.heanet.ie). When finished, the session capture will be analyzed. Windows computers use the FTP client, **`ftp`**, to connect to the FTP server

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

Start a Wireshark capture on the host interface.

Start an FTP connection to the ftp server. Type the command:

```
> ftp ftp.heanet.ie
```

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

Change the FTP directory to `/pub/`

```
ftp> cd pub
ftp> cd putty
```

Terminate the FTP sessions in each command line window with the FTP **`quit`** command:

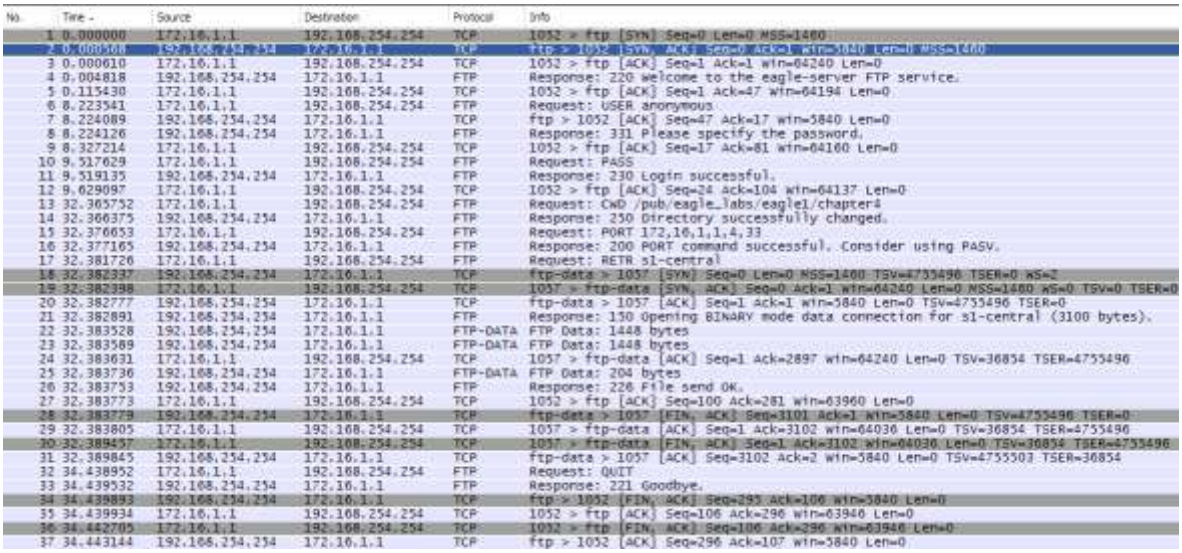
```
ftp> quit
```

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

```
> exit
```

Stop the Wireshark capture.

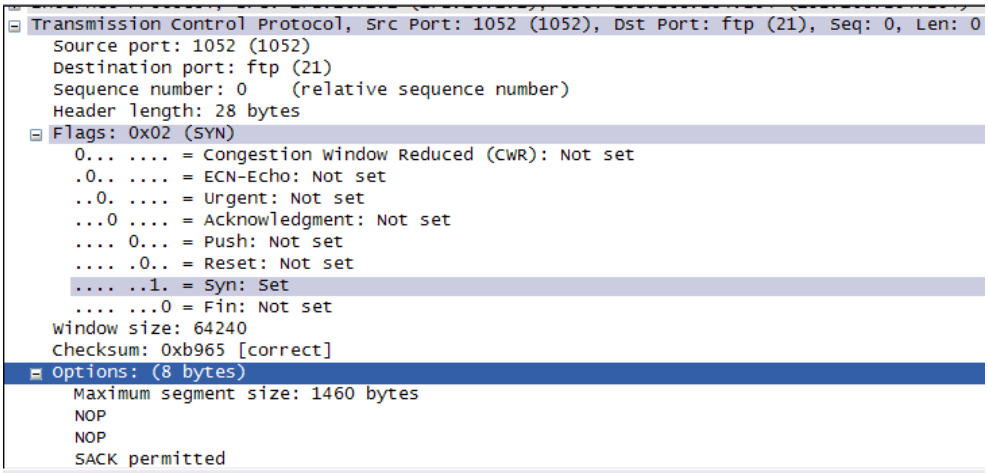
Step 2: Analyze the TCP fields.



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	0.000308	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. To view your FTP session you can highlight any one of the FTP packets and goto Analyze-Follow TCP Stream. What do you notice about the password you typed in?

How is the first datagram in a TCP session identified?

_____

_____

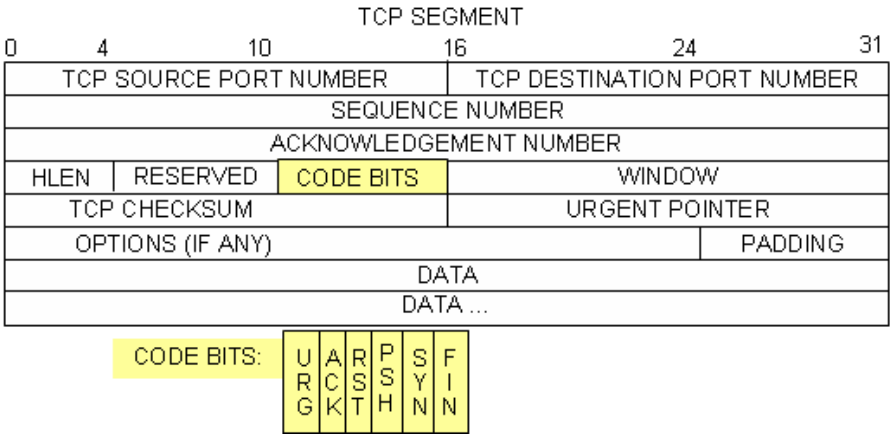


Figure 4. TCP packet fields.

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 host computer to FTP Server (only the SYN bit is set to 1):

Source IP Address: ___.___.___.___	
Destination IP Address: _____	
Source port number: _____	
Destination port number: _____	
Sequence number: _____	
Acknowledgement number: _____	
Header length: _____	
Window size: _____	

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

Source IP Address: _____	
Destination IP Address: ___.___.___.___	
Source port number: _____	
Destination port number: _____	
Sequence number: _____	
Acknowledgement number: _____	
Header length: _____	
Window size: _____	

From host computer to FTP server (only ACK bit is set to 1):

Source IP Address: ___.___.___.___	
Destination IP Address: _____	
Source port number: _____	
Destination port number: _____	
Sequence number: _____	
Acknowledgement number: _____	
Header length: _____	
Window size: _____	

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

_____

_____

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.

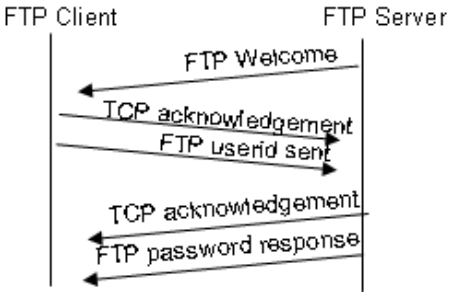


Figure 5. TCP session management.

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 FTP server. This sequence is shown in Figure 5, and is visible in the Wireshark capture.

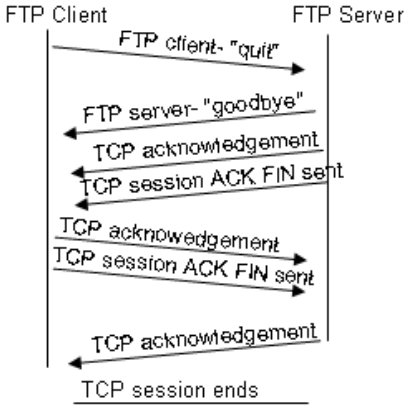


Figure 6. Orderly TCP session termination.

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. Start the TFTP server on your machine to allow a class mate connect to your tftp server.

Start a Wireshark capture, then download a file from your classmates tftp server The command and syntax to perform this is shown below:

>tftp 10.10.100.41 get testfile

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.

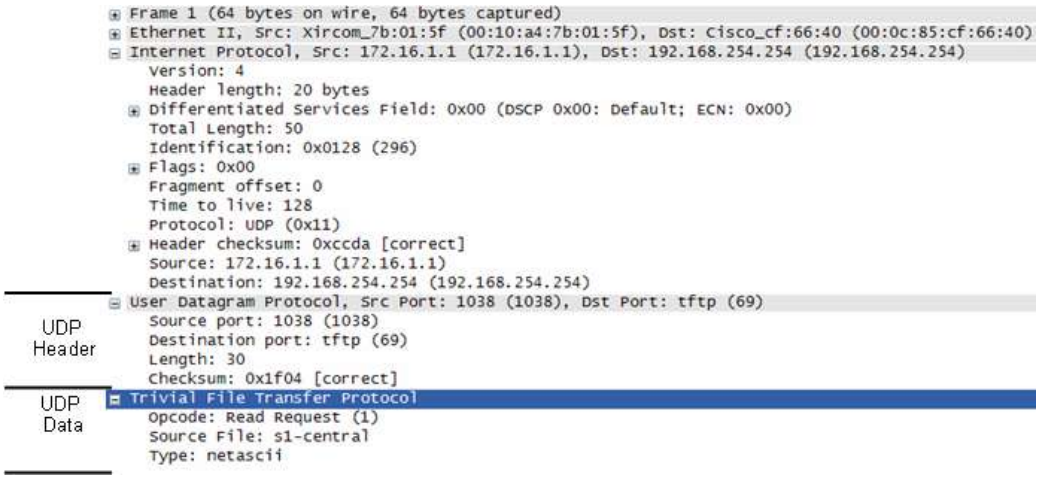


Figure 8. Wireshark capture of a UDP datagram.

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.

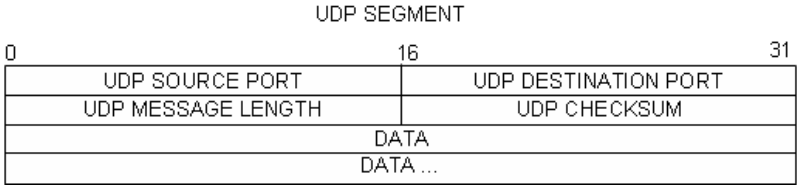


Figure 9. UDP format.

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: __.___.___.___	
Destination IP Address: _____	
Source port number: _____	
Destination port number: _____	
UDP message length: _____	
UDP checksum: _____	

How does UDP verify datagram integrity?

_____

_____

Examine the first packet returned from the TFTP server. Fill in information about the UDP header:

Source IP Address:	
Destination IP Address: ___.___.___.___	
Source port number: _____	
Destination port number: _____	
UDP message length: _____	
UDP checksum: 0x _____	

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. Explore the use of the netstat command to determine what port numbers are in use.

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.