IT 307- Exploring the Networks

LAB HANDOUT-2

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

Introduction to Wireshark:
Wireshark is a widely-used network protocol analyzer that lets you capture and interactively browse the traffic running on a computer network. It can decode various protocols and provides detailed analysis of packets.

Basics of TCP and UDP Captures:
- TCP (Transmission Control Protocol): A connection-oriented protocol that ensures reliable delivery of packets. TCP sessions can be captured to analyze handshakes, data transfer, and terminations.
- UDP (User Datagram Protocol): A connectionless protocol that provides fast but unreliable delivery. UDP captures can be useful for analyzing protocols like DNS, VoIP, and streaming services.

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

TCP Header Fields:
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Reserved
Flags (SYN, ACK, FIN, RST, PSH, URG)
Window Size
Checksum
Urgent Pointer
Options

Steps:

1. **Setup FTP Server and Client:** Ensure that an FTP server is running, and a client is ready to connect.

    Windows:
    Setting Up FTP Server:

    1**. Install Internet Information Services (IIS):**
    - Open Control Panel > Programs and Features > Turn Windows features on or off.
    - Check "Internet Information Services" and "FTP Server."
    - Click OK to install.

2. **Configure FTP Site:**
   - Open IIS Manager.
   - Right-click "Sites" > Add FTP Site.
   - Provide a name and choose a physical path (folder) for your FTP site.
   - Choose "No SSL" if you are testing locally.
   - Assign permissions for the users.

3. **Allow Firewall Rules (if applicable):**
   - Open Windows Firewall.
   - Allow FTP traffic on ports 20 and 21.

Setting Up FTP Client:

1. **Use Built-in Windows FTP Client or Install Third-party Client (e.g., FileZilla):**
   - Open Command Prompt.
   - Type `ftp localhost` and press Enter.
   - Enter the username and password as configured in the FTP server.

2. **Access Files:**
   - Use `get` and `put` commands to download and upload files respectively.

Mac/Linux:

Setting Up FTP Server:

1. **Enable FTP Server (ftpd):**
   - Open Terminal.
   - Type `sudo -s launchctl load -w /System/Library/LaunchDaemons/ftp.plist` and press Enter.
   - Provide administrator password.

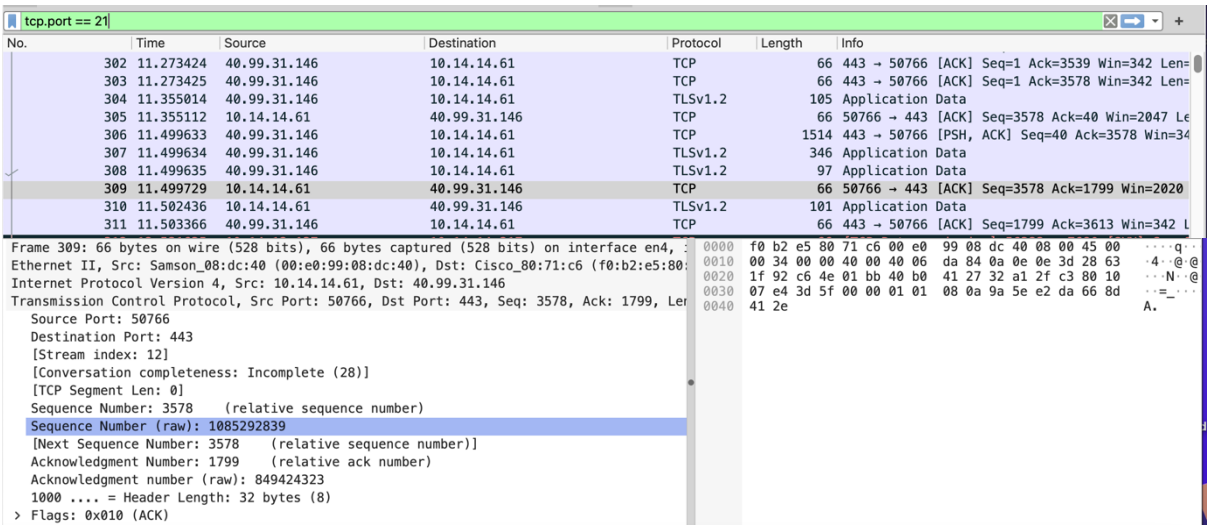
2. **Configure User Permissions:**
   - Choose users and set permissions as needed.

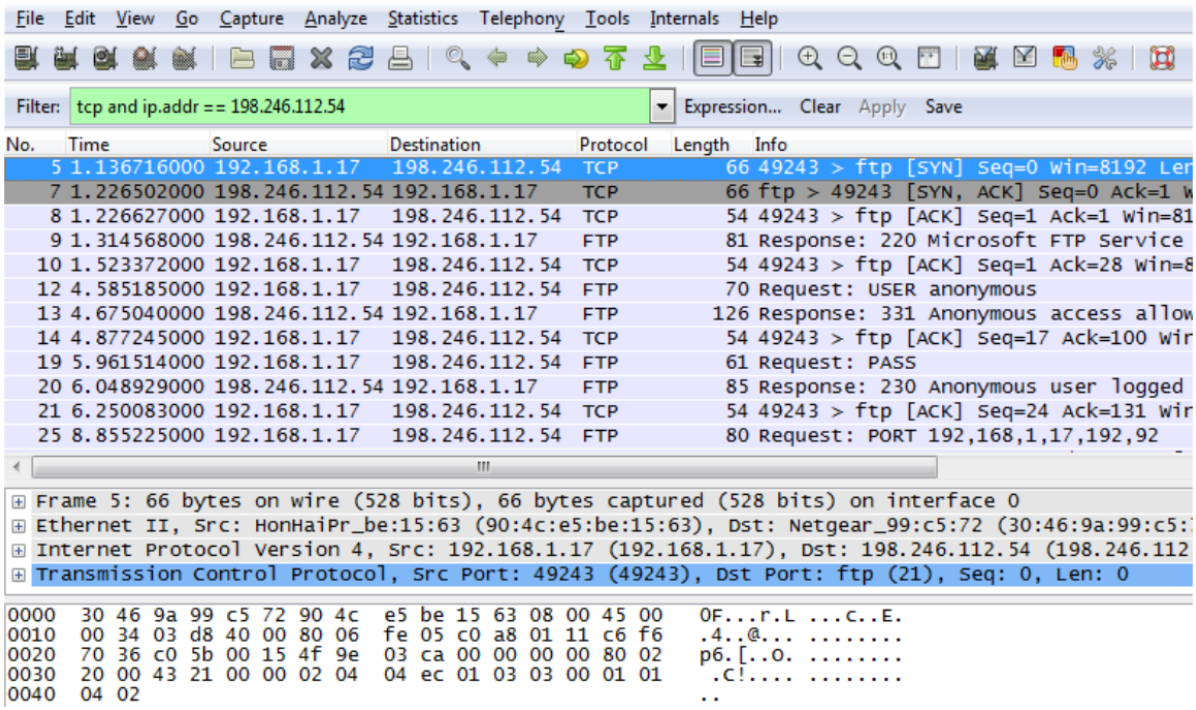
3. **Allow Firewall Rules (if applicable):**
   - Go to System Preferences > Security & Privacy > Firewall.
   - Allow FTP traffic if necessary.

Setting Up FTP Client:
1. Use Built-in macOS FTP Client or Install Third-party Client (e.g., FileZilla):
   - Open Terminal.
   - Type `ftp localhost` and press Enter.
   - Enter the username and password as configured in the FTP server.

2. **Access Files:**
   - Use `get` and `put` commands to download and upload files respectively.

2. **Start Wireshark Capture:** Open Wireshark and start a capture on the interface connected to the network.
3. **Connect to FTP Server:** Perform an FTP transfer between the client and server.
4. **Apply Filter:** In Wireshark, apply the filter tcp.port == 21 to isolate FTP traffic.
5. **Analyze TCP Header:** Click on a TCP packet to analyze the above-mentioned TCP header fields.
6. **Advanced Analysis:** Utilize features such as "Follow TCP Stream" for deeper inspection.





Part 2: UDP Header Fields and Operation (TFTP Session Capture)

Required Tools:

- Wireshark
- TFTP client and server setup

UDP Header Fields:

1. Source Port
2. Destination Port
3. Length
4. Checksum

Steps:

1. Setup TFTP Server and Client: Ensure that a TFTP server is running, and a client is ready to connect.

Windows:
Setting Up TFTP Server:

1. **Install TFTP Server Software (e.g., Tftpd32 or Tftpd64):**

Download and install the software from the official site.
Run the software and choose a directory for file sharing.
Configure Server Settings (Optional):

2. **Set up security, logging, or other specific configurations as needed.**
3. **Allow Firewall Rules (if applicable):**

Open Windows Firewall.
Allow TFTP traffic on port 69.

Setting Up TFTP Client:

1. **Use Built-in Windows TFTP Client:**
Open Command Prompt.
Use tftp command followed by the host (localhost) and operations like get or put. For example: tftp localhost get filename.txt.

Mac/Linux:
Setting Up TFTP Server:
Enable TFTP Server:

1. **Open Terminal.**
Type sudo launchctl load -F /System/Library/LaunchDaemons/tftp.plist and press Enter.
Provide administrator password.
Configure Server Directory:

2. **The default path for TFTP is /private/tftpboot/.**
Use sudo mkdir /private/tftpboot/ to create the directory if it doesn't exist.
Set appropriate permissions with sudo chmod and sudo chown commands.

3. **Allow Firewall Rules (if applicable):**

Go to System Preferences > Security & Privacy > Firewall.
Allow TFTP traffic if necessary.

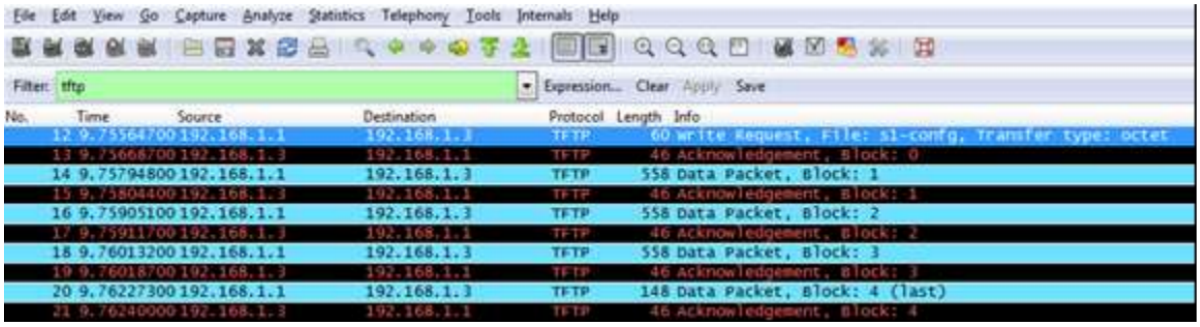
Setting Up TFTP Client:

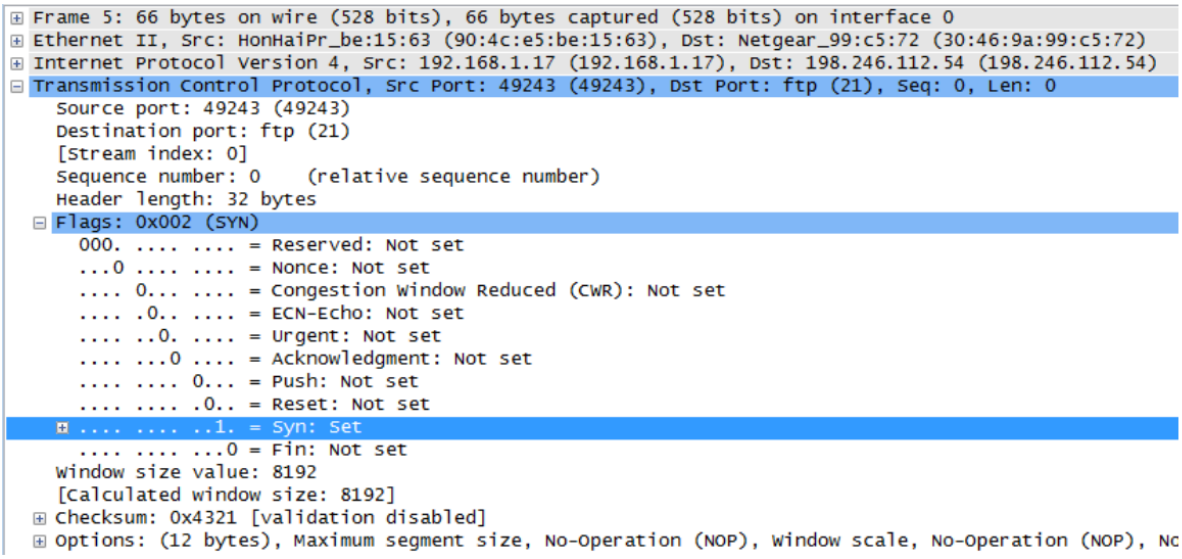
1. **Use Built-in macOS TFTP Client:**

Open Terminal.
Use tftp command followed by the host (localhost) and operations like get or put. For example: tftp localhost get filename.txt.

Note:
1. Setting up a TFTP client and server on your localhost for both Windows and Mac systems is straightforward. It enables a minimalistic and connectionless file transfer mechanism, often used in local network configurations.
2. TFTP lacks security features such as encryption and authentication. It's generally used in controlled environments, and extra caution should be taken if used outside local testing scenarios.

2. Start Wireshark Capture: Open Wireshark and start a capture on the interface connected to the network.
3. Connect to TFTP Server: Perform a TFTP transfer between the client and server.
4. Apply Filter: In Wireshark, apply the filter udp.port == 69 to isolate TFTP traffic.
5. Analyze UDP Header: Click on a UDP packet to analyze the above-mentioned UDP header fields.
6. Advanced Analysis: Utilize features like "Statistics" to view endpoints and conversations.





Filters in Wireshark:
Filters allow you to narrow down the packet view in Wireshark to specific criteria. They can be based on protocols, ports, IP addresses, and various other attributes.

Examples of Filters:

1. **Protocol Filters:**
   -HTTP Traffic: `http`
   - ICMPv6 Traffic:`icmpv6`

2.**Port Filters:**
   - SSH Traffic (TCP Port 22): `tcp.port == 22`
   - Non-DNS Traffic (UDP Port 53):`udp.port != 53`

3. **IP Address Filters:**
   - Packets from Specific Source IP: `ip.src == 192.168.1.10`
   - Packets to Specific Destination IP:`ip.dst == 192.168.2.20`

4. **Logical Operators:**
   - TCP Packets from Specific IP: `tcp && ip.src == 192.168.1.1`
   - ARP or DHCP Packets: `arp || bootp`

5. **Comparison Operators:**
   - TCP Packets with Length > 500: `tcp.len > 500`
   - UDP Packets with Length ≤ 300: `udp.length <= 300`

6. **Expression Filters:**
   - HTTP GET Requests:`http.request.method == "GET"`
   - DNS Queries for Specific Domain:`dns.qry.name contains "google.com"`

7. **Conversation and Flow Filters:**
   - Specific TCP Stream Number 10:`tcp.stream eq 10`
   - Specific UDP Stream Number 5:`udp.stream eq 5`

8. **Frame Filters:**
   - Frames with Length ≥ 1000 Bytes:`frame.len >= 1000`
   -200th Frame:'frame.number == 200`

9. **Custom Filters:**
   - SMTP Traffic with Specific Source Port: `smtp && tcp.srcport == 25`
   - TCP Packets with SYN Flag Set: `tcp.flags.syn == 1`

More examples to try

1. Protocol Filters
You can filter by specific protocol types:
- `tcp`: Filters all TCP packets.
- `udp`: Filters all UDP packets.
- `ip`: Filters all IP version 4 packets.
- `ipv6`: Filters all IP version 6 packets.
- `arp`: Filters all ARP requests and replies.

2. Port Filters
You can filter packets based on port numbers:
- `tcp.port == 80`: Filters TCP packets on port 80.
- `udp.port != 53`: Filters UDP packets not on port 53.

3. IP Address Filters
Filter packets based on IP addresses:
- `ip.src == 192.168.1.1`: Filters packets from a specific source IP.
- `ip.dst == 192.168.1.2`: Filters packets to a specific destination IP.
- `ip.addr == 192.168.1.1`: Filters packets from or to a specific IP address.

4. Logical Operators
Combine filters using logical operators:
- `tcp && ip.src == 192.168.1.1`: Filters TCP packets from a specific IP.
- `udp || arp`: Filters either UDP or ARP packets.

5. Comparison Operators
Utilize comparison operators for more specific filtering:
- `tcp.len > 100`: Filters TCP packets with a length greater than 100 bytes.
- `udp.length <= 200`: Filters UDP packets with a length less than or equal to 200 bytes.

6. Expression Filters
Build complex expressions:
- `http.request.method == "GET"`: Filters HTTP GET requests.
- `dns.qry.name contains "example.com"`: Filters DNS queries containing "example.com."

7. Conversation and Flow Filters
Focus on specific conversations or flows:
- `tcp.stream eq 5`: Filters the specific TCP stream number 5.
- `udp.stream eq 3`: Filters the specific UDP stream number 3.

8. Frame Filters
Analyze packets at the frame level:
- `frame.len >= 200`: Filters frames with a length of 200 bytes or more.
- `frame.number == 100`: Filters the 100th frame.

References

Wireshark Official Documentation- https://www.wireshark.org/docs/
TCP/IP - http://www.tcpipguide.com