**Lab 08**

**4.1.**

**A) On which port number(s) does FTP communicate?**

**20 and 21**

**B) Which command can we use for filtering a given service name?**

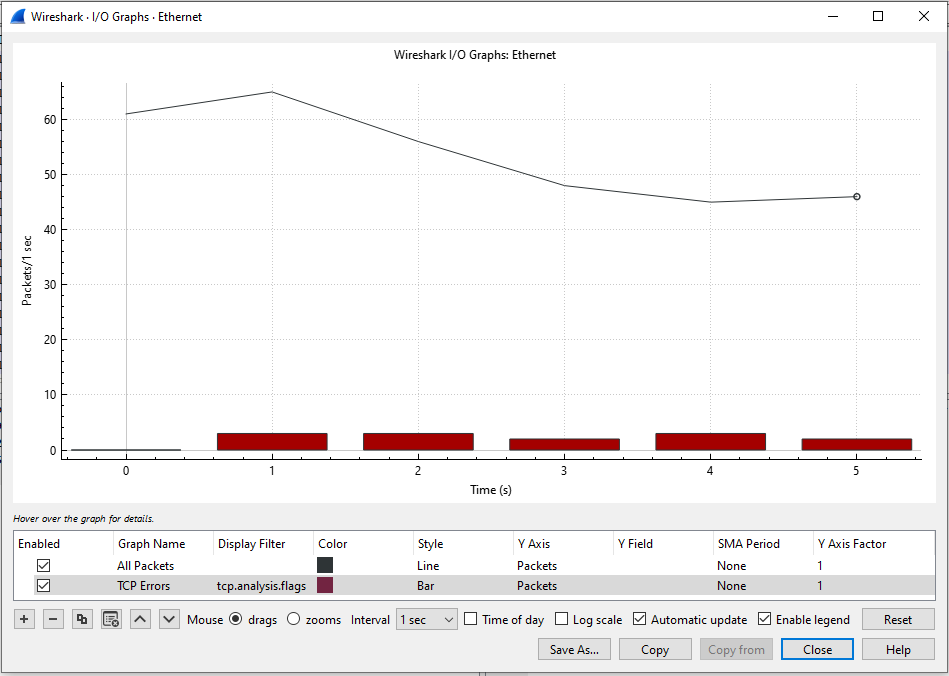
**Grep**

**4.2.**

**A) How many Wireshark filter types do you know? Name them.**

1. **Protocol Filters:**
   * **These filters allow you to focus on specific network protocols. For example:**
     + **ip: Displays all packets containing the IP protocol.**
     + **tcp: Shows only TCP packets.**
     + **udp: Filters UDP packets.**
2. **Field Filters:**
   * **You can filter packets based on specific fields within protocols. Examples include:**
     + **ip.addr == 192.168.1.1: Filters packets with the source or destination IP address of 192.168.1.1.**
     + **tcp.port == 80: Displays packets using port 80 (HTTP).**
3. **Logical Operators:**
   * **Combine filters using logical operators:**
     + **and: Requires both conditions to be true.**
     + **or: Requires at least one condition to be true.**
     + **not: Negates a condition.**
4. **Comparison Operators:**
   * **Use these to compare field values:**
     + **== (equal)**
     + **!= (not equal)**
     + **> (greater than)**
     + **< (less than)**
     + **>= (greater than or equal to)**
     + **<= (less than or equal to)**
5. **Existence Filters:**
   * **Check if a field or protocol exists:**
     + **tr.rif: Filters packets containing the Token-Ring RIF field.**

**B) How can we make a diagram-based statistical report with Wireshark regarding the captured packets?**

1. **Capture File Properties:**
   * **To get general information about the data captured over the network, follow these steps:**
     + **Start Wireshark.**
     + **Click on Statistics in the main menu.**
     + **Choose Capture File Properties.**
   * **In the Capture File Properties window, you’ll find details such as:**
     + **File: Provides file data (filename, path, length, etc.).**
     + **Time: Start time, end time, and duration of capture.**
     + **Capture: Hardware information for the PC where Wireshark is installed.**
     + **Interfaces: Interface information (registry identifier, interface type, packet size limit).**
     + [**Statistics: General capture statistics, including captured and displayed packets1**](https://hub.packtpub.com/statistical-tools-in-wireshark-for-packet-analysis/)**.**
2. **Protocol Hierarchy:**
   * **To analyze the protocol distribution in the captured file, follow these steps:**
     + **Start Wireshark.**
     + **Click on Statistics.**
     + **Choose Protocol Hierarchy.**
   * **The Protocol Hierarchy window displays:**
     + **Protocol: The protocol name.**
     + **Percent Packets: The percentage of protocol packets from the total captured packets.**
     + **Packets: The number of protocol packets.**
     + **Percent Bytes: The percentage of protocol bytes from the total captured packets.**
     + [Bytes: The number of protocol bytes](https://hub.packtpub.com/statistical-tools-in-wireshark-for-packet-analysis/)
3. **Flow Graph:**
   * **The Flow Graph window shows connections between hosts, including packet time, direction, ports, and comments for each captured connection.**
   * [You can filter connections by ICMP Flows, ICMPv6 Flows, UIM Flows, and TCP Flow](https://www.wireshark.org/docs/wsug_html_chunked/ChStatFlowGraph.html)
   * ****

**4.3.**

**a) How many system states does the Finite State Machine (FSM) contain which manages TCP connection?**

**The Finite State Machine (FSM) that manages a TCP connection typically consists of six primary states. These states represent different phases of the connection lifecycle:**

1. **Closed: The initial state when no connection exists.**
2. **Listen: The server is waiting for incoming connection requests.**
3. **Syn-Sent: The client initiates a connection by sending a SYN packet.**
4. **Syn-Received: The server acknowledges the client’s SYN packet and sends its own SYN packet.**
5. **Established: The connection is fully established, and data transfer can occur.**
6. **Close-Wait: The server has received a request to close the connection but is waiting for any remaining data to be sent.**
7. **Last-Ack: The server acknowledges the client’s request to close the connection.**
8. **Time-Wait: A final state where the server waits for any delayed packets before fully closing the connection.**

**B) What is the difference between an active and a passive TCP connection termination?**

1. **Active Termination (Client-Side):**
   * **In an active termination, the client (initiator of the connection) actively initiates the termination process.**
   * **The steps involved in active termination are as follows:**
     1. **The client sends a FIN (Finish) packet to the server, indicating that it has finished sending data.**
     2. **The server acknowledges the FIN packet with an ACK (Acknowledgment).**
     3. **The server may continue to send any remaining data it has.**
     4. **Once the server is done, it sends its own FIN packet to the client.**
     5. **The client acknowledges the server’s FIN packet.**
     6. **The connection transitions to the TIME-WAIT state for a brief period before fully closing.**
2. **Passive Termination (Server-Side):**
   * **In passive termination, the server (receiver of the connection) passively waits for the client to initiate termination.**
   * **The steps involved in passive termination are as follows:**
     1. **The server receives a FIN packet from the client.**
     2. **The server acknowledges the FIN packet with an ACK.**
     3. **The server may continue to send any remaining data it has.**
     4. **Once the server is done, it sends its own FIN packet to the client.**
     5. **The client acknowledges the server’s FIN packet.**
     6. **The connection transitions to the TIME-WAIT state.**
3. **TIME-WAIT State:**
   * **Both active and passive terminations end up in the TIME-WAIT state.**
   * **This state ensures that any delayed packets related to the terminated connection are handled correctly.**
   * **After a brief period (usually a few minutes), the connection is fully closed.**

**4.4**

**A)Can we use netstat command in Windows environment?**

**yes**

**B) How (using which command and what way) can we determine if we have an active web connection with destination host IP address 193.225.40.94?**

**netstat -aon | findstr ":80" (in command prompt)**

**4.5**

**A) Can we use the netcat tool for implementing a chat program?**

1. **Create the Listener:**
   * **On d1, run the following command to set up Netcat as a listener on a specific port (e.g., port 1234):**
   * **nc -nvlp 1234**
     + **Flags used:**
       - **-n: Do not resolve hostnames via DNS.**
       - **-v: Set verbosity level (verbose).**
       - **-l: Bind and listen for incoming connections.**
       - **-p: Specify the source port to use.**
2. **Connect to the Listener:**
   * **On the other device (let’s call it d2), run the following command to connect to the listener on d1 using the IP address you noted down earlier:**
   * **nc <localhost\_IP> 1234**

**Replace <localhost\_IP> with the actual IP address from step 2.**

1. **Chat Away!:**
   * **Now you have a simple chat set up between d1 and d2 using Netcat!**

**B) How could we block the running of such a chat environment in the most simple way?**

1. **Network Firewall or Proxy Rules:**
   * **Configure your network firewall or proxy to block traffic on the specific port used by the chat program (e.g., port 1234 for Netcat).**
   * **Deny incoming and outgoing connections on that port to prevent communication.**
2. **Application Whitelisting/Blacklisting:**
   * **Use application control or whitelisting/blacklisting tools to manage which applications are allowed to run.**
   * **Blacklist Netcat or any other chat-related tools to prevent their execution.**
3. **User Access Control:**
   * **Limit user permissions to execute specific programs.**
   * **Ensure that only authorized users can run chat programs.**

**4.6.**

**A) Can file transfer happen above UDP?**

**yes**

**B) In the case of Netcat-based file transfer is server or client side necessary to implement first? Justify your answer.**

**In the context of Netcat-based file transfer, the server side should be implemented first. Let me explain why:**

1. **Server-Client Model:**
   * **Netcat operates using a client-server model.**
   * **The server listens for incoming connections, while the client initiates connections to the server.**
   * **In file transfer scenarios, the server acts as the receiver of files, and the client sends the files.**
2. **Order of Execution:**
   * **To establish communication, the server must be ready to accept incoming connections.**
   * **If the client is implemented first, it won’t have anywhere to send data until the server is up and running.**
   * **Therefore, implementing the server side first ensures that the communication channel is available when the client connects.**
3. **Steps for Implementation:**
   * **Here’s the typical order of steps:**
     1. **Server Side:**
        + **Set up the server using Netcat to listen on a specific port.**
        + **Example: nc -nvlp 1234**
     2. **Client Side:**
        + **Connect the client to the server using the server’s IP address and port.**
        + **Example: nc <server\_IP> 1234**
     3. **File Transfer:**
        + **Once the connection is established, the client can send files to the server.**
4. **Justification:**
   * **Implementing the server side first ensures that the communication infrastructure is ready before clients attempt to connect.**
   * **It prevents potential issues where clients connect to a non-existent server.**
   * **Additionally, it follows the natural flow of communication establishment.**

**4.7.**

**A) Which option can we use for setting the scanned server side in listening mode?**

**nc -l 4444**

**B) Can we use the netcat tool for scanning several ports which are opened at the same time?**

**nc -zvn google.com 1-100**

**4.8.**

**A) Can we use nmap port for opening?**

**yes**

**B) Which port numbers are open on freemail.hu server?**

**nmap freemai.hu parancs segítségével:**

|  |  |
| --- | --- |
| **Port** | **Státusz** |
| **25/tcp** | **Zárt** |
| **80/tcp** | **Nyitott** |
| **110/tcp** | **Nyitott** |
| **143/tcp** | **Nyitott** |
| **443/tcp** | **Nyitott** |
| **993/tcp** | **Nyitott** |
| **995/tcp** | **Nyitott** |