|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Pratik Pujari** | | |
| **UID no.** | **2020300054** | **Class:** | **Comps C Batch** |
| **Experiment No.** | 4 | | |

|  |  |
| --- | --- |
| **AIM:** | To capture packet using WireShark application |
| **THEORY:** | **What is a network packet?**  A network packet is a small amount of data sent over Transmission Control Protocol/Internet Protocol (TCP/IP) networks. The packet size is around 1.5 kilobytes for Ethernet and 64 KB for IP payloads.  A packet is the unit of data routed between an origin and a destination on the internet or other packet-switched network -- or networks that ship data around in small packets.  **What is a network packet?**  A network packet is a small amount of data sent over Transmission Control Protocol/Internet Protocol (TCP/IP) networks. The packet size is around 1.5 kilobytes for Ethernet and 64 KB for IP payloads.  A packet is the unit of data routed between an origin and a destination on the internet or other packet-switched network -- or networks that ship data around in small packets.  **What are the parts of a network packet?**  Network packets are made up of three different parts: header, payload and trailer. Conceptually, they're like a postal package. In this scenario, the header is the box/envelope, the payload is content and the trailer is the signature.  The header contains instructions related to the data in the packet. These instructions can include the following:   * checksum, which detects errors; * 16-bit identification number; * flags to let a router know if it can fragment a packet; * fragmentation offsets, which reconstruct fragmented packets; * destination address; * number of hops a packet can make; * IP; * length of the packet -- but not always, as some networks have fixed-length packets; * size of the header and payload; * time-to-live; * originating address; * packet number, in relation to the packet sequence; * protocol or what type of packet is transmitted; and * synchronization or the few bits that enable the packet to match up to the network. * The payload is the data within the packet. This is the basic information that the packet delivers to the destination.     **What Is Wireshark?**  Wireshark is a network protocol analyzer, or an application that captures packets from a network connection, such as from your computer to your home office or the internet. Packet is the name given to a discrete unit of data in a typical Ethernet network.  Wireshark is the most often-used packet sniffer in the world. Like any other packet sniffer, Wireshark does three things:   1. **Packet Capture:** Wireshark listens to a network connection in real time and then grabs entire streams of traffic – quite possibly tens of thousands of packets at a time. 2. **Filtering:** Wireshark is capable of slicing and dicing all of this random live data using filters. By applying a filter, you can obtain just the information you need to see. 3. **Visualization:** Wireshark, like any good packet sniffer, allows you to dive right into the very middle of a network packet. It also allows you to visualize entire conversations and network streams.   How to Install Wireshark on Linux  If you have a Linux system, you’d install Wireshark using the following sequence (notice that you’ll need to have root permissions):   * $ sudo apt-get install wireshark * $ sudo dpkg-reconfigure wireshark-common * $ sudo usermod -a -G wireshark $USER * $ newgrp wireshark   Once you have completed the above steps, you then log out and log back in, and then start Wireshark:   * $ wireshark & |
| **EXPERIMENT 1** | |
| **QUESTIONS:** | 1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?     HTTP version:1.1   1. What languages (if any) does your browser indicate that it can accept to the server? In the captured session, what other information (if any) does the browser provide the server with regarding the user/browser?     Language used : English US   1. What is the IP address of your computer?     IP Address: 10. 0. 2.15   1. What is the status code returned from the server to your browser? 2. When was the HTML file that you are retrieving last modified at the server? 3. How many bytes of content are being returned to your browser? 4. By inspecting the raw data in the "packet bytes" pane, do you see any http headers within the data that are not displayed in the "packet details" pane? If so, name one. 5. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET? 6. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell? 7. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header? 8. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain. 9. How many HTTP GET request messages were sent by your browser? 10. How many data-containing TCP segments were needed to carry the single HTTP response? 11. What is the status code and phrase associated with the response to the HTTP GET request? 12. Is there any HTTP header information in the transmitted data associated with TCP segmentation? 13. How many HTTP GET request messages were sent by your browser? To which Internet addresses were these GET requests sent? 14. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain. 15. What is the server’s response (status code and phrase) in response to the initial HTTP GET message from your browser? 16. When your browser sends the HTTP GET message for the second time, what new field is included in the HTTP GET message? 17. What does the "Connection: close" and "Connection: Keep-alive" header field imply in HTTP protocol? When should one be used over the other? |
| **SIMULATION:** |  |
| **OUTPUT TABLE:** |  |
| **RESULT:** | |

|  |  |
| --- | --- |
| **EXPERIMENT 2** | |
| **CALCULATION:** |  |
| **SIMULATION:** |  |
| **OUTPUT TABLE:** |  |
| **RESULT:** | |

|  |  |
| --- | --- |
| **EXPERIMENT 3** | |
| **CALCULATION:** |  |
| **SIMULATION:** |  |
| **OUTPUT TABLE:** |  |
| **RESULT:** | |

|  |  |
| --- | --- |
| **EXPERIMENT 4** | |
| **CALCULATION:** |  |
| **SIMULATION:** |  |
| **OUTPUT TABLE:** |  |
| **RESULT:** | |

|  |  |
| --- | --- |
| **EXPERIMENT 5** | |
| **CALCULATION:** |  |
| **SIMULATION:** |  |
| **OUTPUT TABLE:** |  |
| **RESULT:** | |