Network Traffic Analysis Using Wireshark

# Objective

The objective of this task is to capture live network packets using Wireshark, identify basic protocols and traffic types, and document the findings in a concise report. The deliverables include a packet capture file (.pcap) and a summary of identified protocols.

# Tools Used

* + **Wireshark**: A free and open-source packet analyzer used for network troubleshoot ing, analysis, and protocol development.

# Procedure

The following steps were performed to capture and analyze network traffic:

1. **Installing Wireshark**: Wireshark was downloaded and installed from the official website (<https://www.wireshark.org/>) on a system running a supported operat- ing system (e.g., Windows, macOS, or Linux).
2. **Selecting Network Interface**: Wireshark was launched, and the active network interface (e.g., Wi-Fi or Ethernet) was selected from the interface list to start capturing packets.
3. **Generating Traffic**: To generate network traffic, a web browser was used to visit a website (e.g., [https://www.example.com](https://www.example.com/)), and a ping command was executed in the terminal/command prompt to a server (e.g., ping google.com).
4. **Capturing Packets**: Packet capture was initiated and allowed to run for approx- imately one minute to collect sufficient data.
5. **Stopping Capture**: The capture was stopped after one minute to analyze the collected packets.
6. **Filtering Packets**: Wireshark’s filter bar was used to filter packets by protocols such as HTTP, DNS, TCP, ICMP, and ARP to identify specific traffic types.
7. **Exporting Capture**: The captured packets were saved as a .pcap file (e.g., network*traffic.pcap*)*using SaveAs′′optioninWireshark.***Analyzing Protocols** : *Thecapturedpacketswereanalyzedtoidentif yatl*

1

Network Traffic Analysis with Wireshark 2 of [2](#_bookmark0)

# Findings

The packet capture revealed the following protocols and traffic types:

## 81.. HTTP (Hypertext Transfer Protocol):

* Observed during web browsing to [https://www.example.com](https://www.example.com/).
* Packets showed HTTP GET requests and responses, typically over TCP port 80 (or port 443 for HTTPS).
* Example: A GET request to retrieve the homepage of the website was cap- tured, with a response containing HTML content.

## DNS (Domain Name System):

* + Observed when resolving domain names (e.g., [www.example.com)](http://www.example.com/) to IP ad- dresses.
  + Packets showed DNS queries and responses over UDP port 53.
  + Example: A query for [www.example.com](http://www.example.com/) returned an IP address such as

93.184.216.34.

## ICMP (Internet Control Message Protocol):

* + Observed during the execution of the ping command to google.com.
  + Packets showed ICMP Echo Request and Echo Reply messages.
  + Example: An ICMP Echo Request was sent to 8.8.8.8, with a corresponding Echo Reply received.

# Packet Capture Details

* + **File Name**: network*traffic.pcap***Capture Duration** : *Approximately*60*seconds*
  + **Total Packets Captured**: Approximately 1,000 packets (varies based on network activity)
  + **Network Interface**: Wi-Fi (or Ethernet, depending on the system)
  + **Protocols Identified**: HTTP, DNS, ICMP, TCP, ARP

# Summary

The network traffic capture using Wireshark successfully identified multiple protocols, including HTTP, DNS, and ICMP, among others. The process involved installing Wire- shark, capturing packets on an active network interface, generating traffic through web browsing and ping commands, and analyzing the captured data. Filters were applied to isolate specific protocols, and the capture was exported as a .pcap file for further anal- ysis. The findings demonstrate the presence of common network protocols used in web browsing and diagnostics, providing insights into network communication patterns.