



IT PROJECT DOCUMENTATION

DNS and HTTP Analysis with
Wireshark – Network Analysis
Project

MORGAN BURERA

JUNE 2025

Google IT Support
Professional Certificate

Ubuntu, Terminal



Table of Contents

1. Project Overview
 2. Environment Setup
 3. Test Set up and Process
 4. Capture and Analysis
 5. Resolution / Outcome
 6. Skills Learned / Demonstrated
 7. Personal Enhancement
 8. Video Recording
-

This document is part of a personal project portfolio developed during the Google IT Support Certification. All simulations and analyses were performed in a controlled lab environment. These projects serve as a complement to the course and provide an initial hands-on experience applying its concepts to real-world scenarios

I. Project Overview

As part of my hands-on training in IT Support and Networking, the objective of this project was to observe and analyze basic network communications using **Wireshark**. I focused on **DNS resolution** and **HTTP traffic**, two core elements in most end-user connectivity issues.

II. Environment Setup

Component	Details
OS	Ubuntu 22.04 (Virtual Machine)
Network Type	NAT via interface `enp0s3`
Capture Tools	Wireshark + Tshark (CLI)
Server IP	[dig], [curl], [tshark]
Target Webtsites	google.com, example.com

III. Test Setup and Process

- Identified active network interfaces with ``tshark -D``.
- Chose the one with Internet access: **enp0s3**.
- Launched packet capture with:

```
sudo tshark -i enp0s3 -f "udp port 53 or tcp port 80" -w dns_http.pcapng
```

- In a second terminal, triggered network traffic with:

```
dig google.com  
curl http://example.com
```

- Stopped the capture and analyzed the file using **Wireshark GUI**.

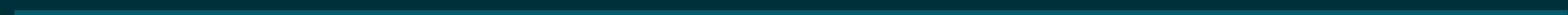
4. Capture and Analysis

Wireshark Filter used :

``dns || http``

Key Findings:

- Clear DNS queries to **Google's public DNS servers (8.8.8.8)**.
- Resolution of ``google.com`` visible with ``A`` record returned.
- -HTTP GET request to ``example.com``, with **``status code 200 OK``**.
- Proper TCP 3-way handshake before HTTP exchange.



V/ Resolution / Outcome

No issues were simulated in this project — the goal was to observe **expected behavior** of DNS and HTTP traffic.

However, this capture can serve as a baseline for comparison in future projects (e.g., failed DNS resolution, HTTP timeouts, latency issues, etc.).

VI/ Key Takeaways & Skills Demonstrated

- Usage of **Tshark** for command-line packet capture
 - **Interface diagnosis** using `tshark -D`` and `ip a``
 - Generation of DNS and HTTP traffic via `dig`` and `curl``
 - Filtered packet analysis using **Wireshark GUI**.
 - Understanding of **network layer*** and typical web browsing flow
-

VIII/ Optional Enhancements, Reflection

During the DNS/HTTP traffic capture with tshark, the command `dig google.com` initially failed. After inspection, I realized the system was using a local DNS resolver (127.0.0.53), which did not forward queries correctly in this setup. To resolve the issue, I manually updated the nameserver to 1.1.1.1 in the configuration, which restored proper DNS resolution and allowed tshark to capture the expected DNS and HTTP exchanges.

For successful 'dig' commands and 'curl', the issue has been resolved and you can see these on the 5th and 6th projects results.

Video Recording Link

<https://youtu.be/YIMlvxDIRil>