PCAP Traffic Analysis Report Using PyShark

1. Introduction

This report documents the analysis of network traffic captured using **Wireshark** and saved as a **PCAP file**, which was later examined using a custom Python script built with the **PyShark** library. The goal of this analysis is to extract and review **DNS queries**, **HTTP requests**, and **TCP sessions** in order to identify potentially **suspicious or noteworthy network activity**.

The analyzed traffic was captured from a live network interface and exported from Wireshark into a .pcap file for offline inspection.

---

2. Objective

The main objectives of this analysis are:

- To parse DNS traffic and identify queried domain names.
- To inspect HTTP requests for visible methods, hosts, and requested resources.
- To enumerate TCP sessions to understand communication flows between hosts.
- To demonstrate how PyShark can be used for basic network forensics and traffic inspection.

---

3. Tools & Technologies Used

- **Wireshark** – Used to capture live network traffic and export it as a PCAP file.
- **Python 3**
- **PyShark** – A Python wrapper for TShark (Wireshark CLI).
- **TShark** – Backend packet analysis engine used by PyShark.

---

4. Methodology

The analysis process follows these steps:

1. Capture network traffic using Wireshark.
2. Save the capture as a .pcap file.

3. Load the PCAP file into Python using PyShark.
4. Apply protocol-based display filters (DNS, HTTP, TCP).
5. Extract relevant fields from each protocol.
6. Print findings to the console for review.

---

5. Script Overview

The script consists of four main functions:

5.1 DNS Traffic Analysis (parse_dns)

- Applies the dns display filter.
- Extracts queried domain names (qry_name).
- Stores all DNS queries found in the capture.

Purpose:
Identify domain lookups which may indicate:

- Command-and-control communication
- Suspicious or unknown domains
- DNS tunneling behavior (in advanced cases)

5.2 HTTP Traffic Analysis (parse_http)

- Applies the http display filter.
- Extracts:
    - HTTP request method
    - Host header
    - Requested URI

Purpose:
Detect:

- Cleartext HTTP communication
- Suspicious endpoints

- Unusual request paths or hosts
- Possible data exfiltration over HTTP

5.3 TCP Session Enumeration (parse_tcp_sessions)

- Applies the tcp display filter.
- Extracts:
  - Source IP and port
  - Destination IP and port
- Builds a readable session format.

Purpose:
Understand:

- Communication relationships between hosts
- Potential unauthorized connections
- Lateral movement patterns

5.4 Main Analysis Function (analyze_pcap)

- Coordinates all parsing functions.
- Prints categorized results:
  - DNS queries
  - HTTP requests
  - TCP sessions
- Gracefully handles missing protocol fields.

---

6. Input Data

- **PCAP File:** sample_traffic.pcap
- **Source:** Captured directly from Wireshark

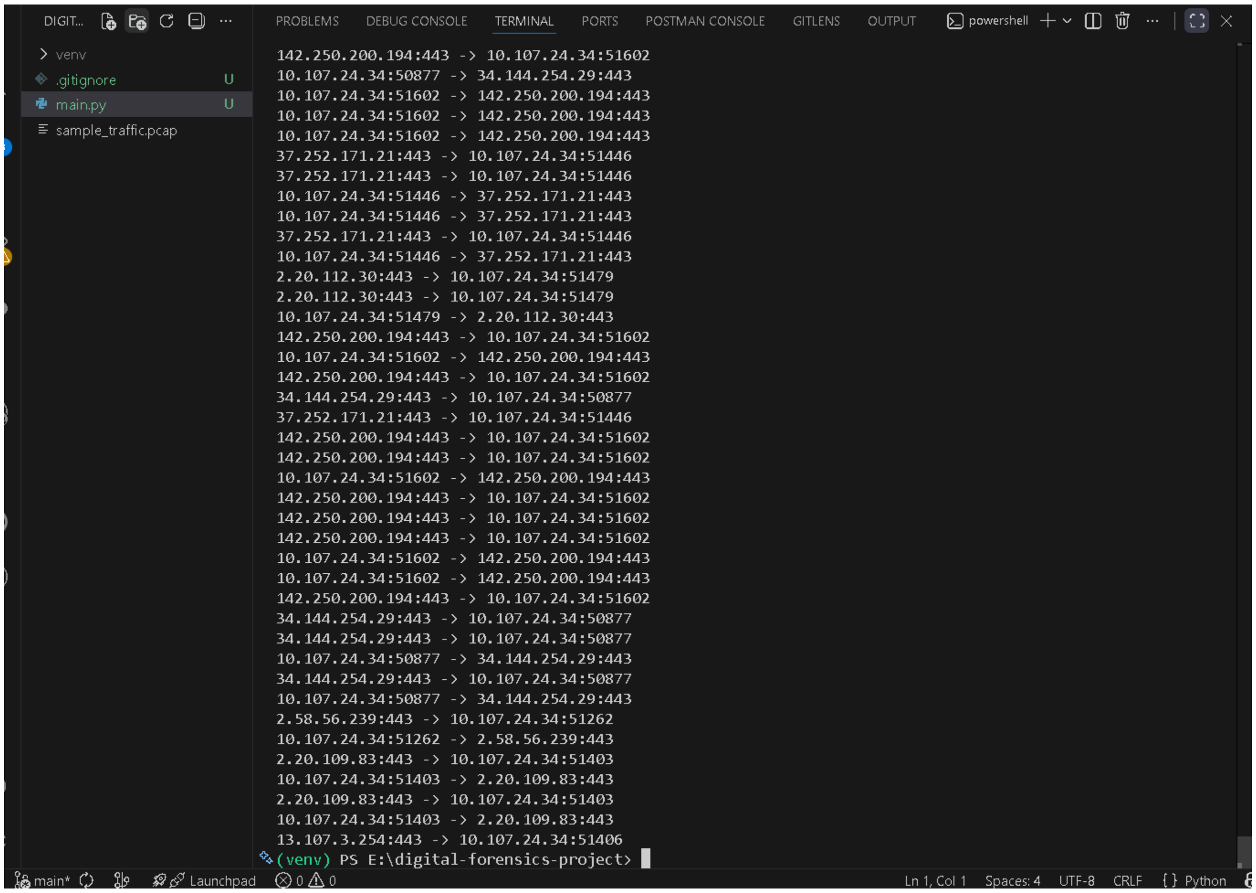
- **Traffic Type:** Real network traffic (not simulated)

---

7. Results & Findings

7.1 DNS Findings

- All detected DNS queries are printed for inspection.
- The script does **not** automatically classify domains as malicious.
- Any suspicious behavior must be manually assessed based on:
    - Unknown domains
    - High-frequency queries
    - Unusual TLDs

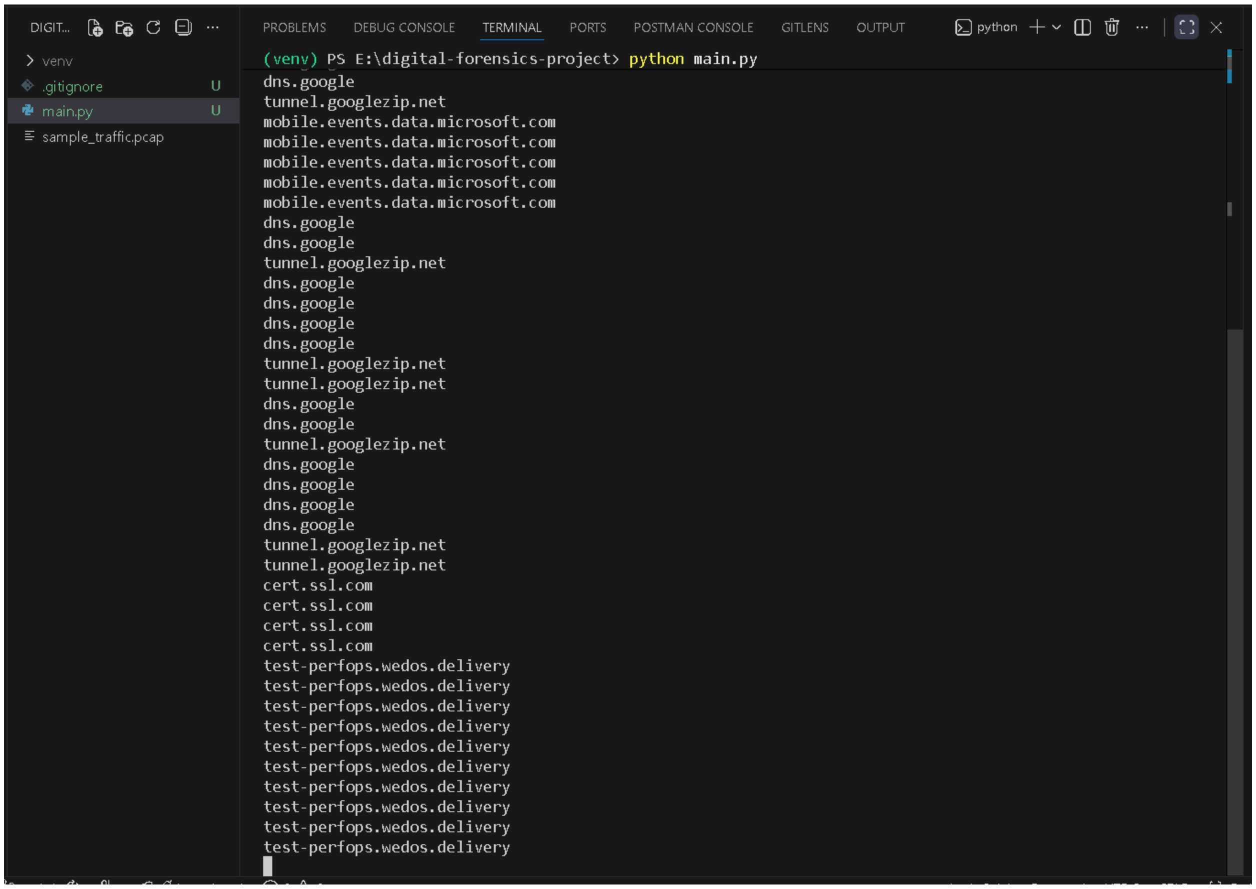


```
142.250.200.194:443 -> 10.107.24.34:51602
10.107.24.34:50877 -> 34.144.254.29:443
10.107.24.34:51602 -> 142.250.200.194:443
10.107.24.34:51602 -> 142.250.200.194:443
10.107.24.34:51602 -> 142.250.200.194:443
37.252.171.21:443 -> 10.107.24.34:51446
37.252.171.21:443 -> 10.107.24.34:51446
10.107.24.34:51446 -> 37.252.171.21:443
10.107.24.34:51446 -> 37.252.171.21:443
37.252.171.21:443 -> 10.107.24.34:51446
10.107.24.34:51446 -> 37.252.171.21:443
2.20.112.30:443 -> 10.107.24.34:51479
2.20.112.30:443 -> 10.107.24.34:51479
10.107.24.34:51479 -> 2.20.112.30:443
142.250.200.194:443 -> 10.107.24.34:51602
10.107.24.34:51602 -> 142.250.200.194:443
142.250.200.194:443 -> 10.107.24.34:51602
34.144.254.29:443 -> 10.107.24.34:50877
37.252.171.21:443 -> 10.107.24.34:51446
142.250.200.194:443 -> 10.107.24.34:51602
142.250.200.194:443 -> 10.107.24.34:51602
10.107.24.34:51602 -> 142.250.200.194:443
142.250.200.194:443 -> 10.107.24.34:51602
142.250.200.194:443 -> 10.107.24.34:51602
142.250.200.194:443 -> 10.107.24.34:51602
10.107.24.34:51602 -> 142.250.200.194:443
10.107.24.34:51602 -> 142.250.200.194:443
142.250.200.194:443 -> 10.107.24.34:51602
34.144.254.29:443 -> 10.107.24.34:50877
34.144.254.29:443 -> 10.107.24.34:50877
10.107.24.34:50877 -> 34.144.254.29:443
34.144.254.29:443 -> 10.107.24.34:50877
10.107.24.34:50877 -> 34.144.254.29:443
2.58.56.239:443 -> 10.107.24.34:51262
10.107.24.34:51262 -> 2.58.56.239:443
2.20.109.83:443 -> 10.107.24.34:51403
10.107.24.34:51403 -> 2.20.109.83:443
2.20.109.83:443 -> 10.107.24.34:51403
10.107.24.34:51403 -> 2.20.109.83:443
13.107.3.254:443 -> 10.107.24.34:51406
(venv) PS E:\digital-forensics-project>
```

7.2 HTTP Findings

- HTTP requests are displayed in clear text.
- Reveals:
    - Request methods (GET, POST, etc.)
    - Target hosts
    - Requested resources

⚠️ **Security Note:**
The presence of HTTP traffic indicates **unencrypted communication**, which may expose credentials or sensitive data.

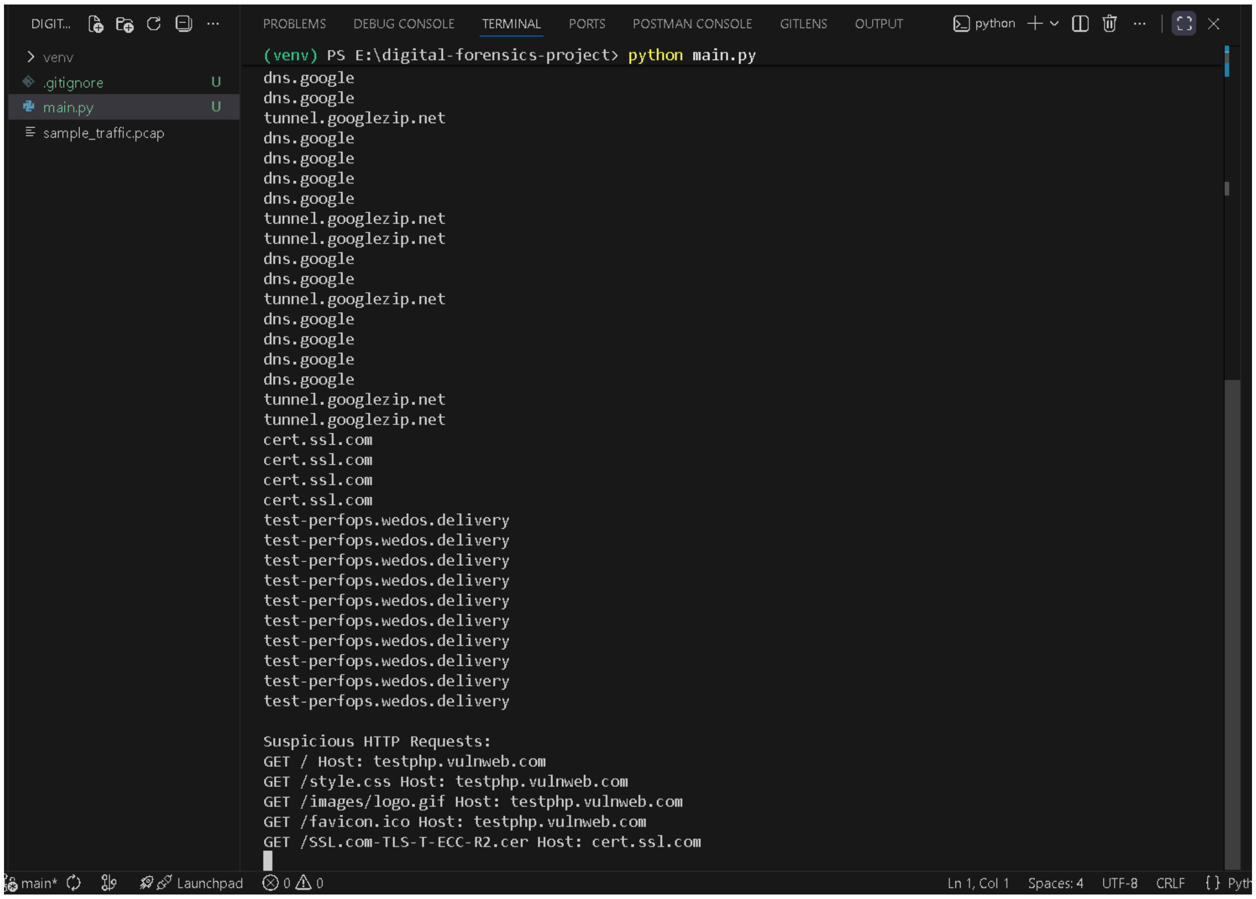


```
(venv) PS E:\digital-forensics-project> python main.py
dns.google
tunnel.googlezip.net
mobile.events.data.microsoft.com
mobile.events.data.microsoft.com
mobile.events.data.microsoft.com
mobile.events.data.microsoft.com
mobile.events.data.microsoft.com
dns.google
dns.google
tunnel.googlezip.net
dns.google
dns.google
dns.google
dns.google
tunnel.googlezip.net
tunnel.googlezip.net
dns.google
dns.google
tunnel.googlezip.net
dns.google
dns.google
dns.google
dns.google
tunnel.googlezip.net
tunnel.googlezip.net
cert.ssl.com
cert.ssl.com
cert.ssl.com
cert.ssl.com
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
```

7.3 TCP Session Findings

- Displays all observed TCP communication paths.
- Useful for:
    - Mapping network activity

- ○ Identifying unexpected external connections
- ○ Detecting unusual port usage



```
(venv) PS E:\digital-forensics-project> python main.py
dns.google
dns.google
tunnel.googlezip.net
dns.google
dns.google
dns.google
dns.google
tunnel.googlezip.net
tunnel.googlezip.net
dns.google
dns.google
tunnel.googlezip.net
dns.google
dns.google
dns.google
dns.google
tunnel.googlezip.net
tunnel.googlezip.net
cert.ssl.com
cert.ssl.com
cert.ssl.com
cert.ssl.com
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery
test-perfops.wedos.delivery

Suspicious HTTP Requests:
GET / Host: testphp.vulnweb.com
GET /style.css Host: testphp.vulnweb.com
GET /images/logo.gif Host: testphp.vulnweb.com
GET /favicon.ico Host: testphp.vulnweb.com
GET /SSL.com-TLS-T-ECC-R2.cer Host: cert.ssl.com
```

---

8. Limitations

- No automated threat intelligence or reputation checks.
- HTTPS traffic contents are not visible.
- No anomaly detection or behavior scoring.
- All results require **manual interpretation**.

---

9. Conclusion

This script provides a **foundational approach to network traffic analysis** using Python and PyShark. It successfully extracts and presents DNS queries, HTTP requests, and TCP session data from a Wireshark-captured PCAP file.

While basic in nature, this approach is effective for:

- Educational purposes
- Initial forensic triage
- Understanding network behavior

Further enhancements could include:

- IOC matching
- HTTPS SNI extraction
- Statistical anomaly detection
- Exporting results to structured reports (JSON/CSV)
