Network Traffic Analysis Report

Task 5: Packet Capture and Protocol Identification Using Wireshark

Kali Linux Virtual Machine Environment

Analyst: FIBIN MN
Date: December 20, 2025
Tool: Wireshark 4.6.0
Environment: Kali Linux (Virtual Machine)
Capture Interface: eth0
Capture File: kali_network_capture.pcap

1. Executive Summary

This report documents a controlled network traffic capture and analysis exercise performed on a Kali Linux virtual machine using Wireshark. The objective was to capture live network packets, identify common network protocols, and demonstrate fundamental packet analysis skills. The capture successfully identified four major protocols: ICMP, DNS, TCP, and TLS, representing diagnostic, name resolution, transport, and security layer communications respectively.

TOTAL PACKETS CAPTURED	CAPTURE DURATION
1,139	~90s

PROTOCOLS IDENTIFIED	NETWORK INTERFACE
4	eth0

2. Methodology

2.1 Environment Setup

Wireshark was configured on Kali Linux with appropriate user permissions for packet capture. The primary network interface (eth0) was selected for monitoring all traffic passing through the VM's network adapter.

```
$ sudo usermod -aG wireshark $USER
$ wireshark
```

2.2 Traffic Generation

To ensure meaningful capture data, traffic was intentionally generated using two methods:

- ICMP traffic generated via ping utility targeting google.com
- HTTP/HTTPS traffic generated via Firefox browser visiting example.com and google.com

```
$ ping google.com -c 5
```

2.3 Capture Process

The capture ran for approximately 60 seconds to collect a representative sample of network activity without excessive noise. No capture filters were applied to ensure complete visibility of all protocols.

3. Protocol Analysis

3.1 ICMP (Internet Control Message Protocol)

Protocol Overview

ICMP is a network layer protocol used for diagnostic and control purposes, primarily implementing the ping utility for reachability testing.

Observed Traffic

Field	Value	Description
Type	8 (Request), 0 (Reply)	Echo request and echo reply
Source IP	192.168.135.130	Kali Linux VM
Destination IP	142.251.222.142	Google server
TTL (outgoing)	64	Linux default
TTL (incoming)	128	Windows-based server response
Sequence Numbers	1/256, 2/512, 3/768, 4/1024, 5/1280	Sequential packet identification

Key Findings

- All five echo requests received corresponding replies, indicating 100% packet delivery
- Round-trip time varied between 40-50ms, typical for internet routing
- TTL differences reveal operating system information about remote hosts
- Data payload contained 40 bytes in each ICMP packet

3.2 DNS (Domain Name System)

Protocol Overview

DNS operates at the application layer to resolve human-readable domain names into IP addresses that computers use for network communication.

Observed Traffic

Query Name	Query Type	Response	Server
ogads-pa.clients6.google.com	A (IPv4)	Multiple A records	192.168.135.2
fonts.gstatic.com	A (IPv4)	142.250.67.35	192.168.135.2
play.google.com	A (IPv4)	142.251.221.206	192.168.135.2

Key Findings

- DNS queries used UDP port 53 for efficient connectionless communication
- Standard query format observed with proper question and answer sections
- Multiple A record responses for load-balanced Google services
- Response times typically under 50ms, indicating local DNS caching
- Browser generated DNS lookups for embedded resources (fonts, scripts)

3.3 TCP (Transmission Control Protocol)

Protocol Overview

TCP is a transport layer protocol providing reliable, connection-oriented communication between network applications through a three-way handshake mechanism.

Three-Way Handshake Analysis

Packet	Flags	Seq Number	Ack Number	Purpose
1	SYN	0 (relative)	-	Client initiates connection
2	SYN, ACK	0 (relative)	1	Server acknowledges and responds
3	ACK	1	1	Client confirms connection established

Connection Details

Parameter	Value
Source Port	56906 (ephemeral/dynamic)
Destination Port	80 (HTTP), 443 (HTTPS)
Window Size	64240 bytes
MSS (Maximum Segment Size)	1460 bytes

Key Findings

- Clean three-way handshakes observed with no retransmissions
- Window scaling enabled for efficient throughput on modern networks
- Selective acknowledgment (SACK) permitted for improved error recovery
- Proper connection termination with FIN packets observed
- TCP Keep-Alive packets detected maintaining idle connections

3.4 TLS (Transport Layer Security)

Protocol Overview

TLS operates at the session/presentation layer to provide encrypted communication over TCP connections, ensuring confidentiality and integrity of web traffic.

TLS Handshake Observed

Handshake Step	Description
Client Hello	Client proposes TLS version, cipher suites, and random value
Server Hello	Server selects TLS version and cipher suite
Certificate	Server presents X.509 certificate for authentication
Key Exchange	Encrypted session keys negotiated
Application Data	Encrypted HTTP traffic transmitted

Key Findings

- TLS 1.3 negotiated for modern security standards
- All HTTP traffic encrypted, no plaintext credentials visible
- Certificate validation occurred transparently
- Application data fully encrypted - only metadata visible
- Perfect forward secrecy ensured through ephemeral key exchange

4. Traffic Statistics

4.1 Protocol Distribution

Protocol	Packet Count	Percentage	Primary Usage
TCP	~498	43.7%	Web browsing, reliable data transfer
TLS/HTTPS	~380	33.4%	Encrypted web traffic
DNS	~82	7.2%	Domain name resolution
ICMP	10	0.9%	Network diagnostics (ping)
Other	~169	14.8%	Background system traffic

4.2 IP Address Analysis

IP Address	Type	Description
192.168.135.130	Source	Kali Linux VM (local machine)
192.168.135.2	Gateway/DNS	VM host DNS resolver
142.251.x.x	Destination	Google infrastructure
34.107.221.82	Destination	Google Cloud Platform
104.18.27.120	Destination	Cloudflare CDN

5. Security Observations

5.1 Encryption Status

- All web traffic observed used HTTPS/TLS encryption
- No plaintext HTTP credentials or sensitive data visible
- DNS queries transmitted in cleartext (potential privacy concern)
- No suspicious or malicious traffic patterns detected

5.2 Network Behavior

- Normal TCP connection establishment and teardown
- Appropriate use of ephemeral ports for client connections
- No port scanning or reconnaissance activity
- Standard browser user-agent strings observed

6. Wireshark Filter Commands Used

```
# Display only ICMP traffic
icmp

# Display only DNS traffic
dns

# Display only TCP traffic
tcp

# Display only TLS/encrypted traffic
tls

# Display traffic to/from specific IP
ip.addr == 142.251.222.142
```

```
# Display specific TCP port
tcp.port == 443
```

7. Lessons Learned

7.1 Technical Skills Acquired

- Proficiency in Wireshark interface and packet inspection tools
- Understanding of protocol layering (OSI/TCP-IP model)
- Ability to identify protocols by header characteristics
- Recognition of normal vs. anomalous traffic patterns
- Practical application of display filters for traffic isolation

7.2 Protocol Understanding

- TCP reliability mechanisms (handshake, ACKs, retransmissions)
- DNS query/response structure and caching behavior
- ICMP's role in network diagnostics and error reporting
- TLS encryption protecting application data from inspection

7.3 Real-World Applications

- Network troubleshooting and performance analysis
- Security incident investigation and forensics
- Protocol compliance testing and validation
- Application behavior verification during development

8. Conclusion

> **Summary:** This packet capture and analysis exercise successfully demonstrated fundamental network traffic analysis skills using Wireshark on Kali Linux. Four major protocols (ICMP, DNS, TCP, TLS) were identified, analyzed, and documented with specific packet-level details.
>
> **Key Achievement:** The analysis went beyond simple protocol identification to examine packet structure, connection establishment, encryption mechanisms, and traffic patterns. This provided practical insight into how modern networks operate at the packet level.
>
> **Practical Value:** Understanding packet-level network behavior is fundamental for cybersecurity professionals, network administrators, and security analysts. This exercise builds the foundation for advanced topics including intrusion detection, malware analysis, and network forensics.

Next Steps: Future analysis could expand to include malformed packets, network anomalies, encrypted tunnel protocols (VPN), wireless traffic capture, or application-specific protocol analysis (SSH, FTP, SMTP).

9. References and Resources

- Wireshark Official Documentation: https://www.wireshark.org/docs/
- RFC 792 - Internet Control Message Protocol (ICMP)
- RFC 1035 - Domain Names - Implementation and Specification (DNS)
- RFC 793 - Transmission Control Protocol (TCP)
- RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3

Report Generated: December 20, 2025

Confidentiality: Educational Use Only