Traffic Capture Lab

CYBR3010

Professor: Sam El’Awour

Gregory Stephens

October 14, 2025

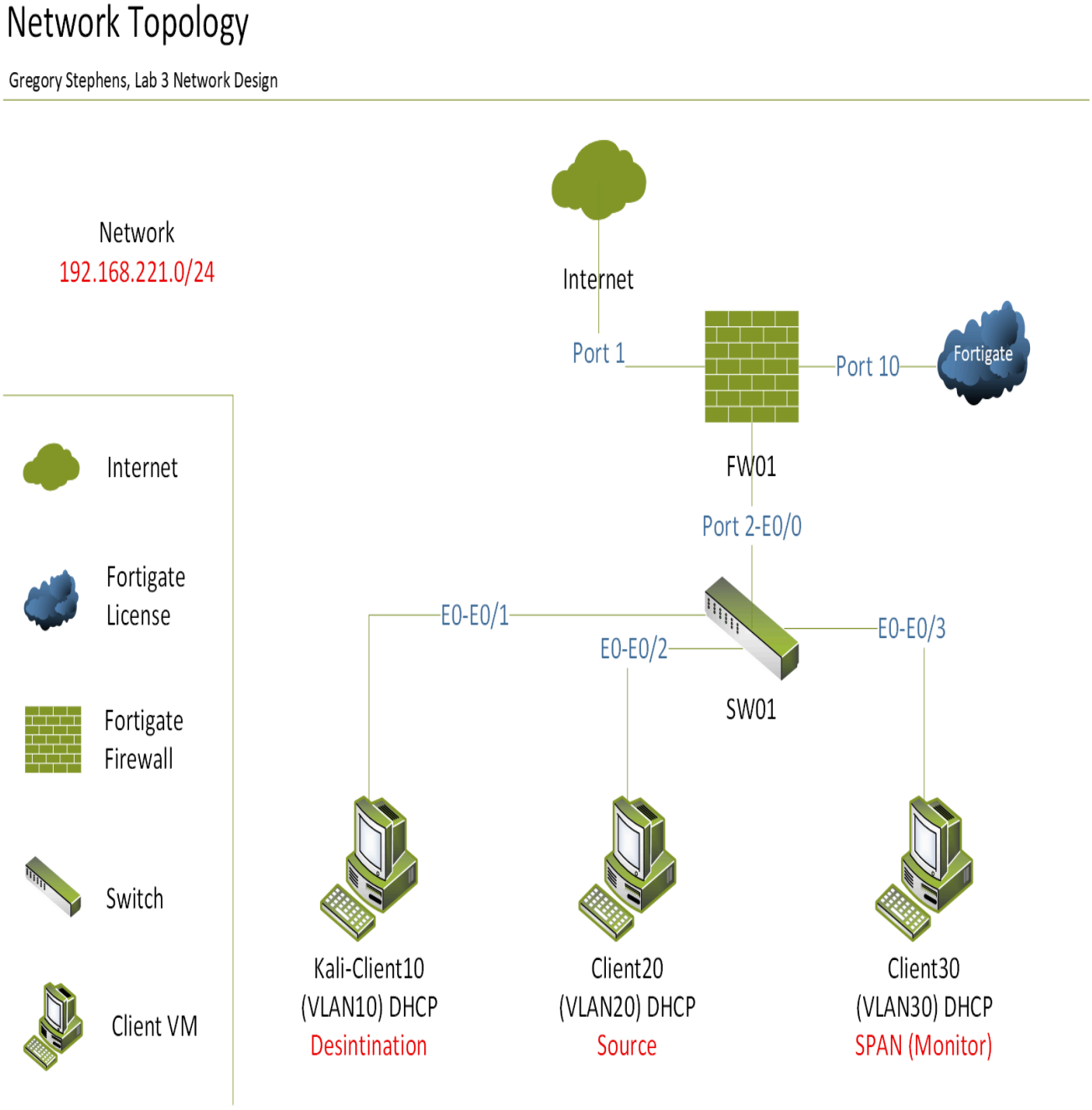
Introduction

This lab demonstrates the placement and use of a protocol analyzer to capture and analyze switched network traffic at multiple points in a small, VLAN-segmented lab network. The environment is hosted in VMware and consists of a single managed switch **(SW01)**, a firewall gateway **(FW01)** that performs inter-VLAN routing and access control, and three virtual machines: **Kali (VLAN10, 192.168.10.50)**, **Client20 (VLAN20, 192.168.20.50)**, and **Client30 (VLAN30, 192.168.30.50)**. **Client30** is configured as the **SPAN (mirror)** destination so it can passively capture all traffic on the trunk between **SW01** and **FW01**; **Client20** and **Kali** are used as the active source and destination to generate and observe traffic.

The primary objective is to capture the same flows simultaneously at the source, on the mirrored trunk **(SPAN)**, and at the destination to demonstrate switched-network behavior, packet visibility differences, and protocol interactions. Wireshark was used on all three VMs to record **ICMP (ping)**, **TCP (HTTP on a temporary Python server)**, and **UDP (DNS)** traffic. Troubleshooting steps were performed to resolve service binding and firewall policy issues so that **TCP** sessions could be established and examined.

This report includes a network diagram, configuration evidence for SPAN, step-by-step capture points, packet analysis of the **TCP three-way handshake and TCP four-way teardown**, a **TTL/MAC** comparison across capture points, and traffic generation logs **(DNS and HTTP bursts)**. All captures are saved and named to map directly to the capture points and traffic types; these files are provided with the submission to allow independent verification and further analysis.

Network Diagram (Visio)



3. SPAN Configuration and Verification

How this screenshot was captured:

On SW01, the following commands were used:

monitor session 1 source interface e0/0 both
monitor session 1 destination interface e0/3
show monitor session 1

Screenshots were taken from the SW01 console during configuration and verification.

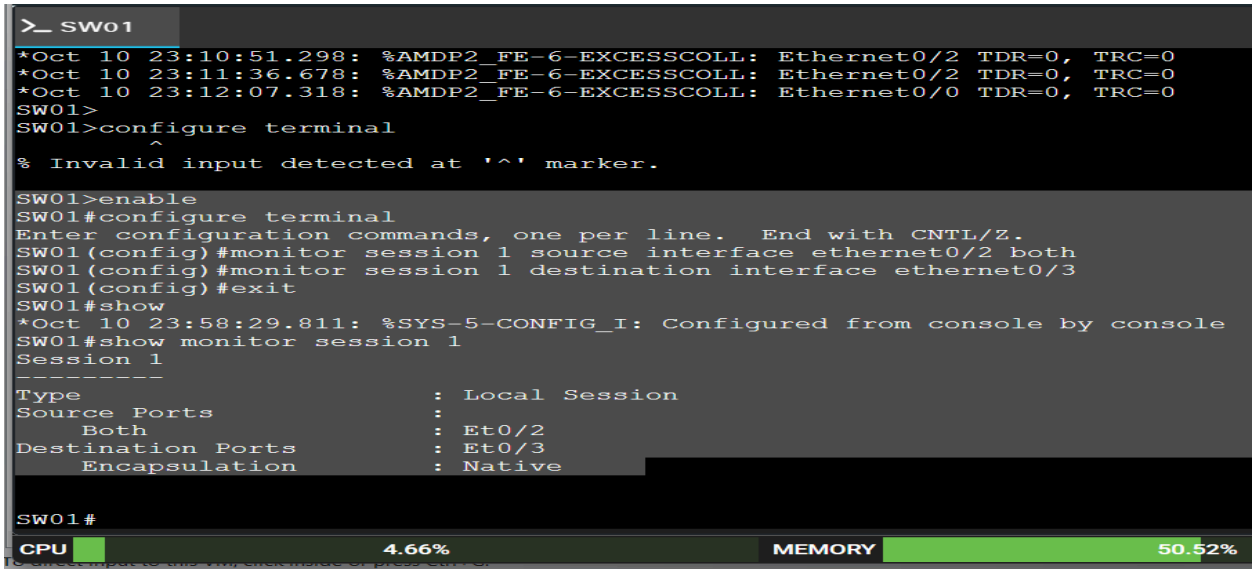


Fig02_SPAN_Config

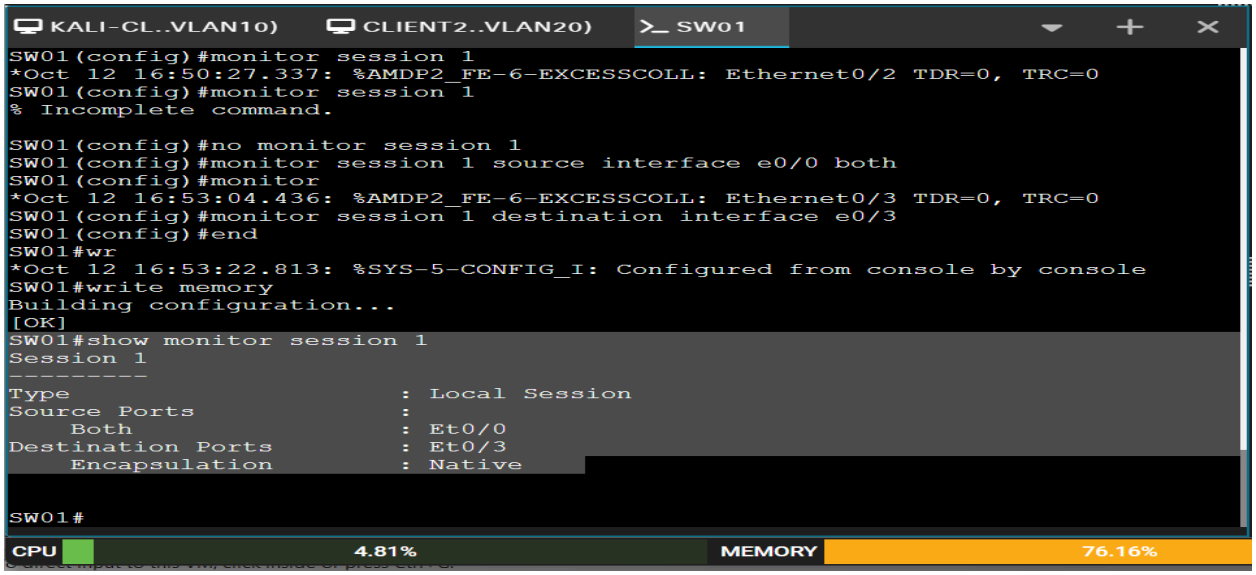


Fig03_SPAN_Verify

4. Connectivity Verification

How this screenshot was captured:

Before beginning traffic analysis, Test-NetConnection was run on Client20 to confirm network reachability.

Kali's HTTP server was started to verify TCP access through the firewall.
Commands used:

- Kali: **sudo python3 -m http.server 8080 --bind 192.168.10.50**
- Client20: **Test-NetConnection 192.168.10.50 -Port 8080**

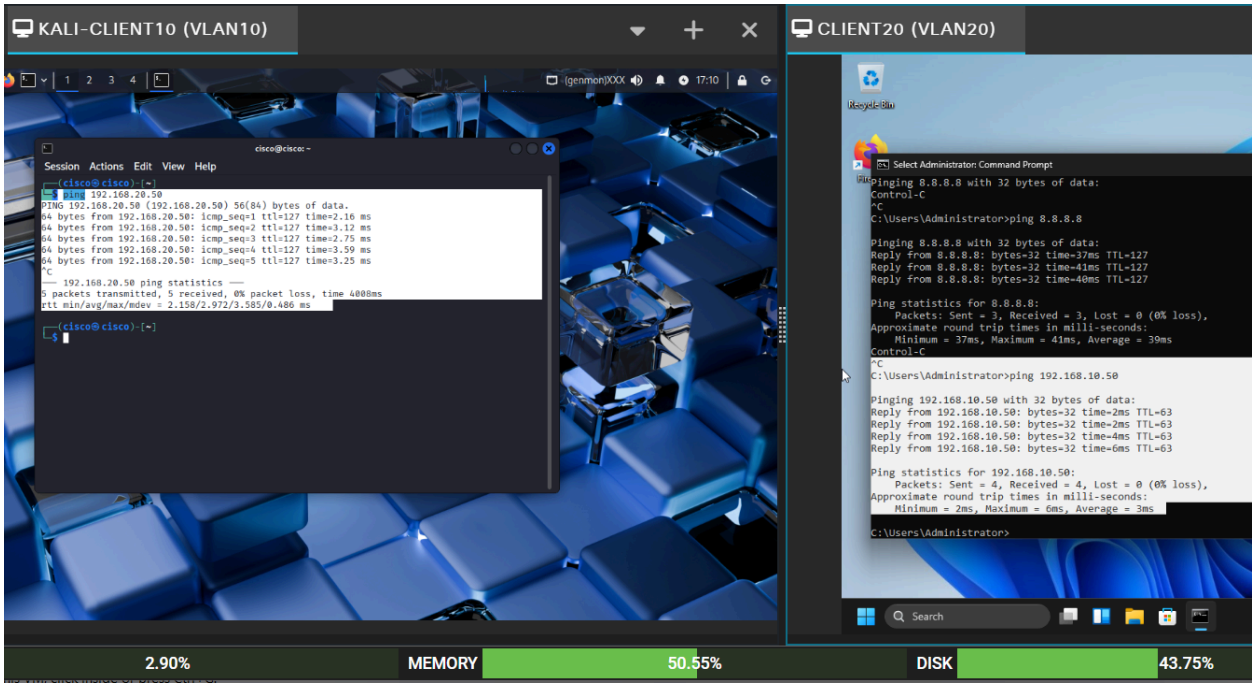


Fig01_Connectivity_Test

5. ICMP (Ping) Capture

How this screenshot was captured:

Wireshark was started on all three VMs. From Client20:

ping 192.168.10.50

Display filter applied:

icmp && ip.addr == 192.168.10.50 && ip.addr == 192.168.20.50

Screenshots were taken after several pings to confirm traffic flow.

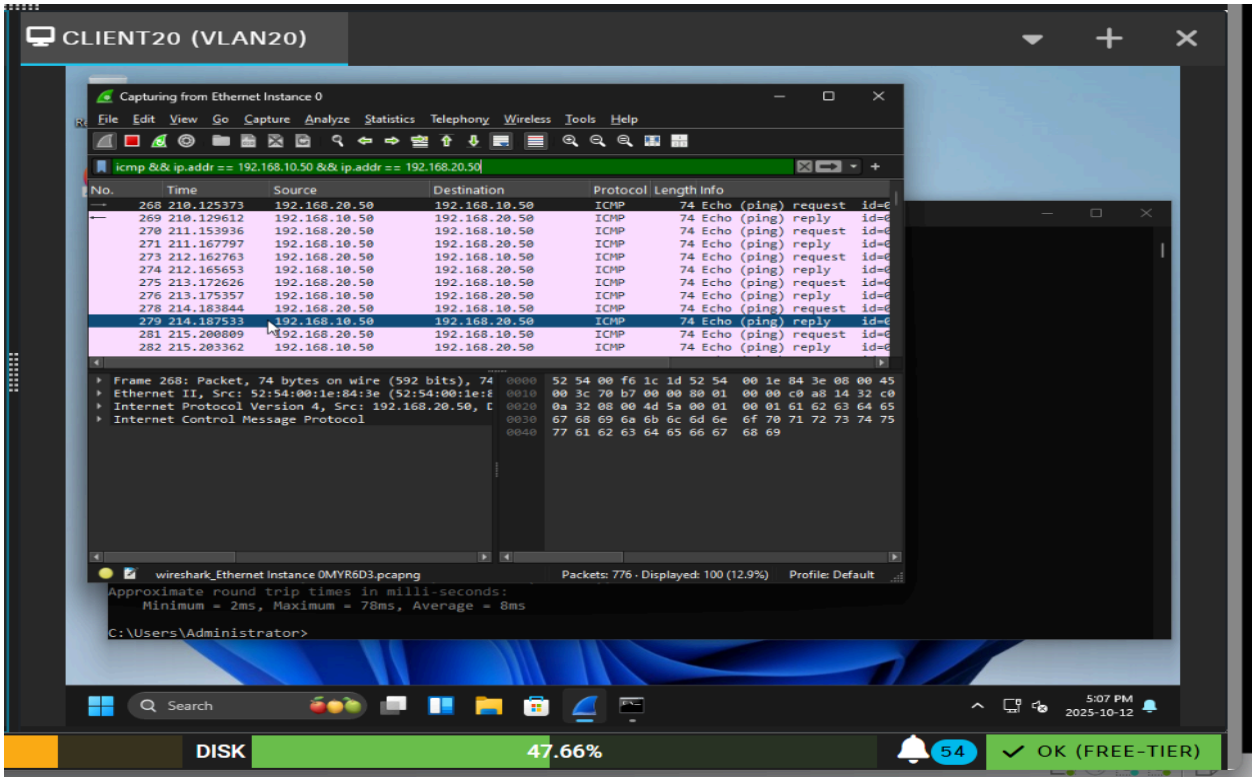


Fig04_ICMP_Source

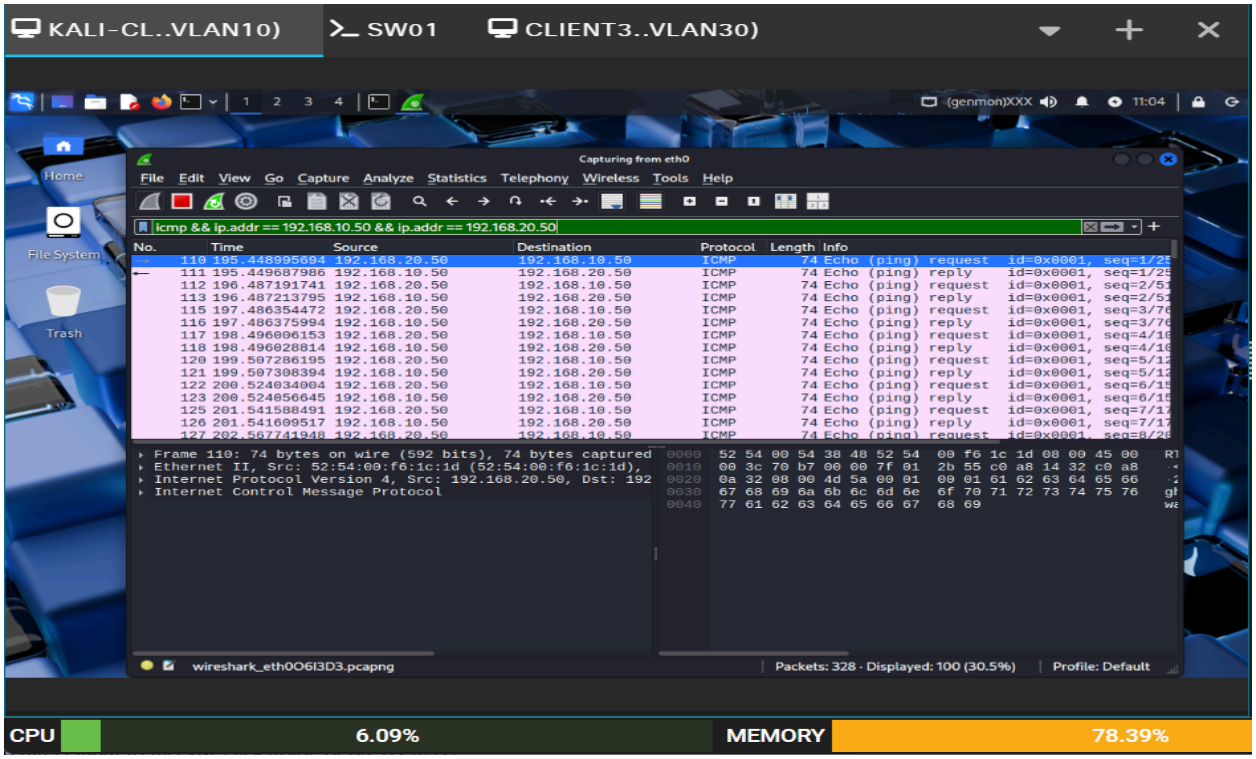


Fig05_ICMP_Destination

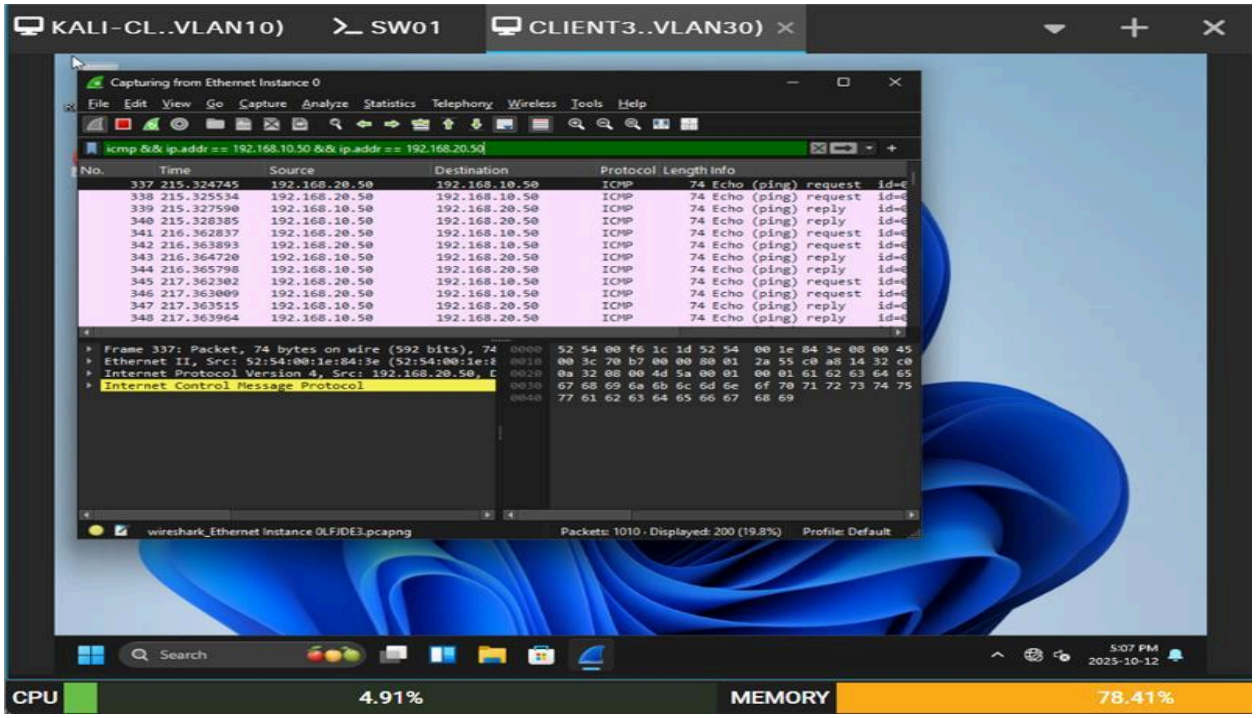


Fig06_ICMP_SPAN

Explanation:

ICMP packets verify that VLAN routing and SPAN mirroring work correctly. Echo requests and replies were visible on all capture points.

Capture Files:
CAP-Source_Client20_ICMP.pcapng
CAP-SPAN_Client30_ICMP.pcapng
CAP-Destination_Kali_ICMP.pcapng

6. TCP Three-Way Handshake

How this screenshot was captured:

After verifying TCP success, Wireshark captured HTTP traffic using:

tcp && ip.addr==192.168.10.50 && ip.addr==192.168.20.50 && tcp.port==8080

Packets were identified as:

- **SYN** → Client20 to Kali
- **SYN/ACK** → Kali to Client20
- **ACK** → Client20 to Kali

The first SYN packet was expanded in the *"Transmission Control Protocol"* section to show flags and TCP options.

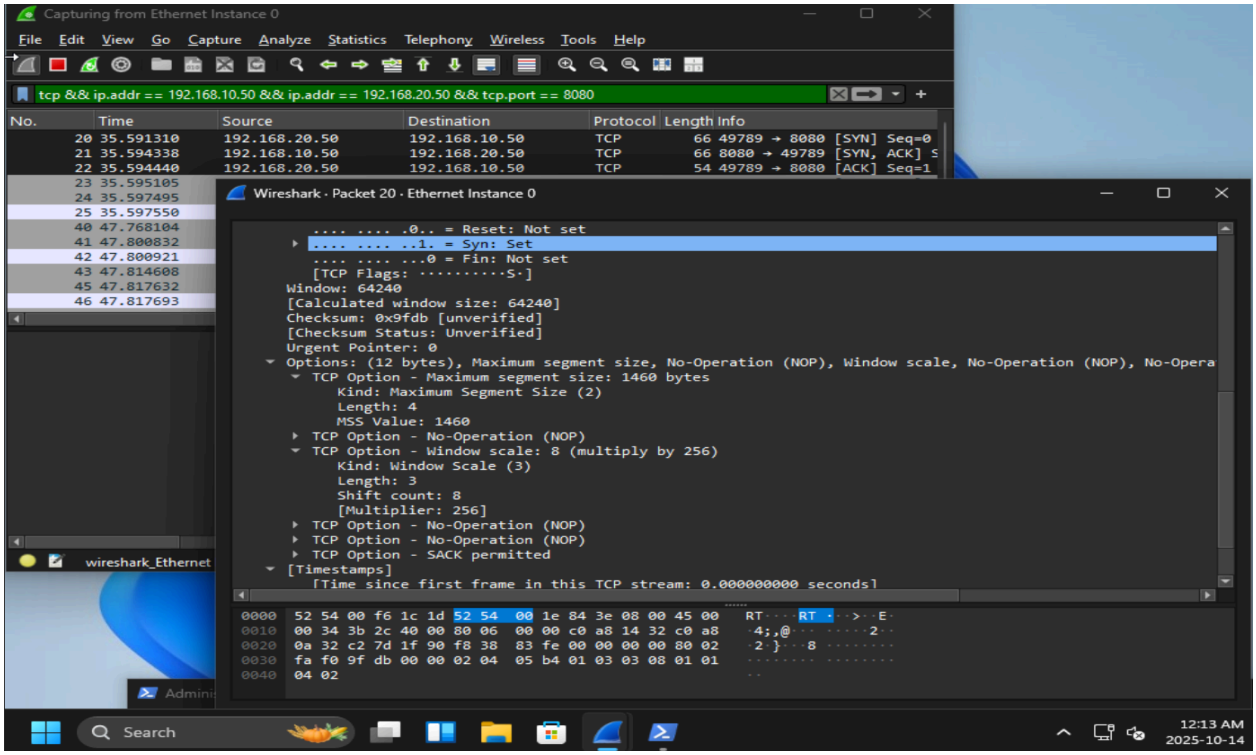


Fig07_TCP_3WH_Source

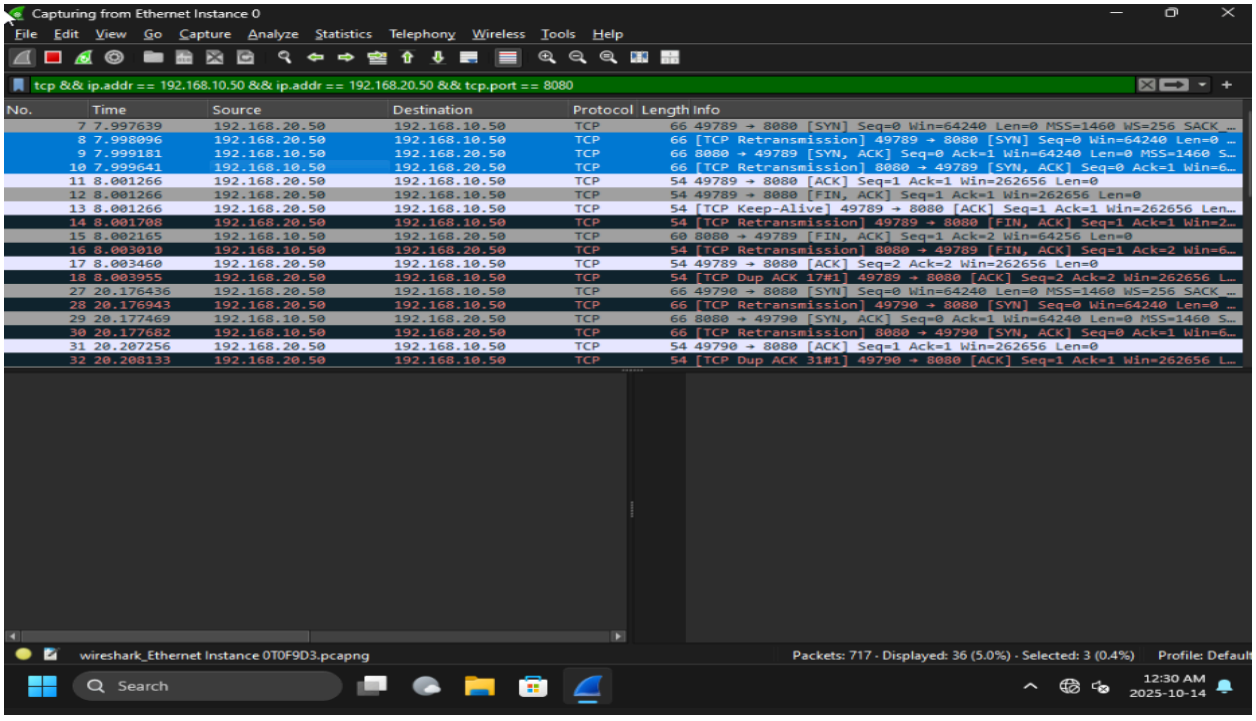


Fig08_TCP_3WH_SPAN

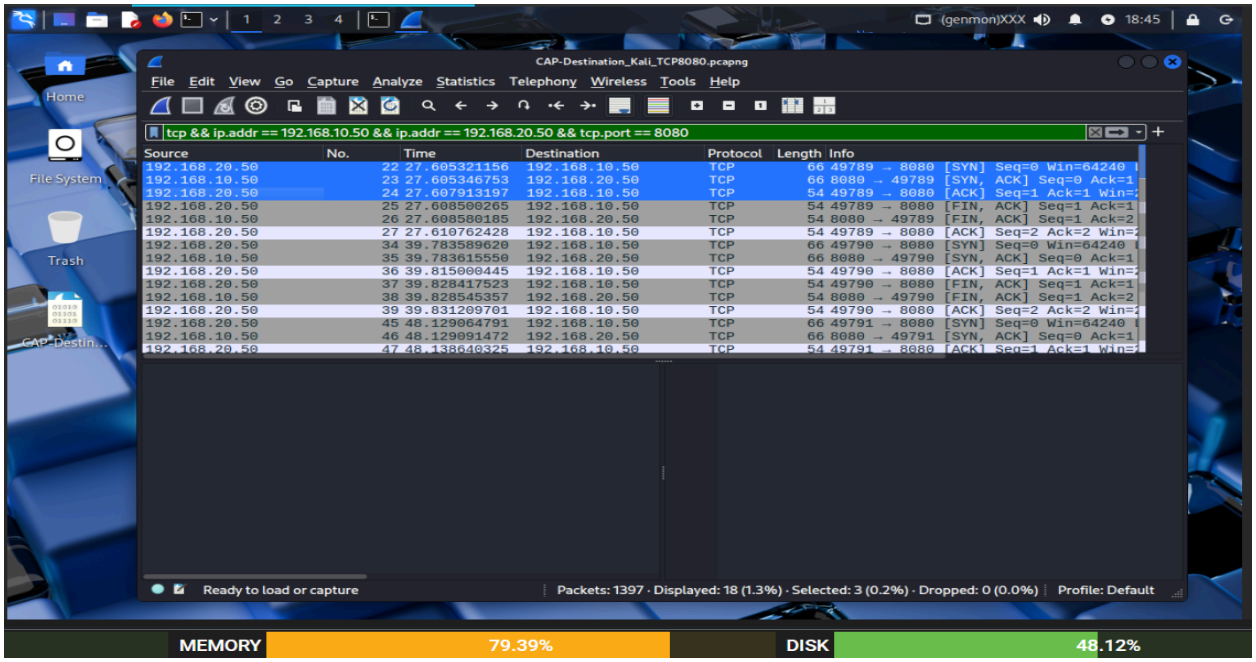


Fig09_TCP_3WH_Destination

Explanation:

The three packets form the TCP handshake that synchronizes sequence numbers. TCP Options observed: MSS 1460, Window Scale 8, SACK Permitted, and Timestamps.

Capture Files:
CAP-Source_Client20_TCP8080.pcapng
CAP-SPAN_Client30_TCP8080.pcapng
CAP-Destination_Kali_TCP8080.pcapng

7. TCP Four-Way Teardown

How this screenshot was captured:

After completing the connection, the client closed the session (browser or PowerShell stopped). The final four packets in the TCP stream were captured, showing:

FIN/ACK → ACK → FIN/ACK → ACK.

Explanation:

This teardown confirms that each side closed its connection gracefully. It demonstrates proper TCP session termination between VLANs through FW01.

Capture Files:
CAP-Source_Client20_TCP8080.pcapng
CAP-SPAN_Client30_TCP8080.pcapng
CAP-Destination_Kali_TCP8080.pcapng

8. DNS (UDP) Traffic

How this screenshot was captured:

From Client20: *(Powershell)*

nslookup nait.ca
nslookup google.com
nslookup yahoo.com

Wireshark filter:

dns

DNS requests and responses were visible on all three VMs.

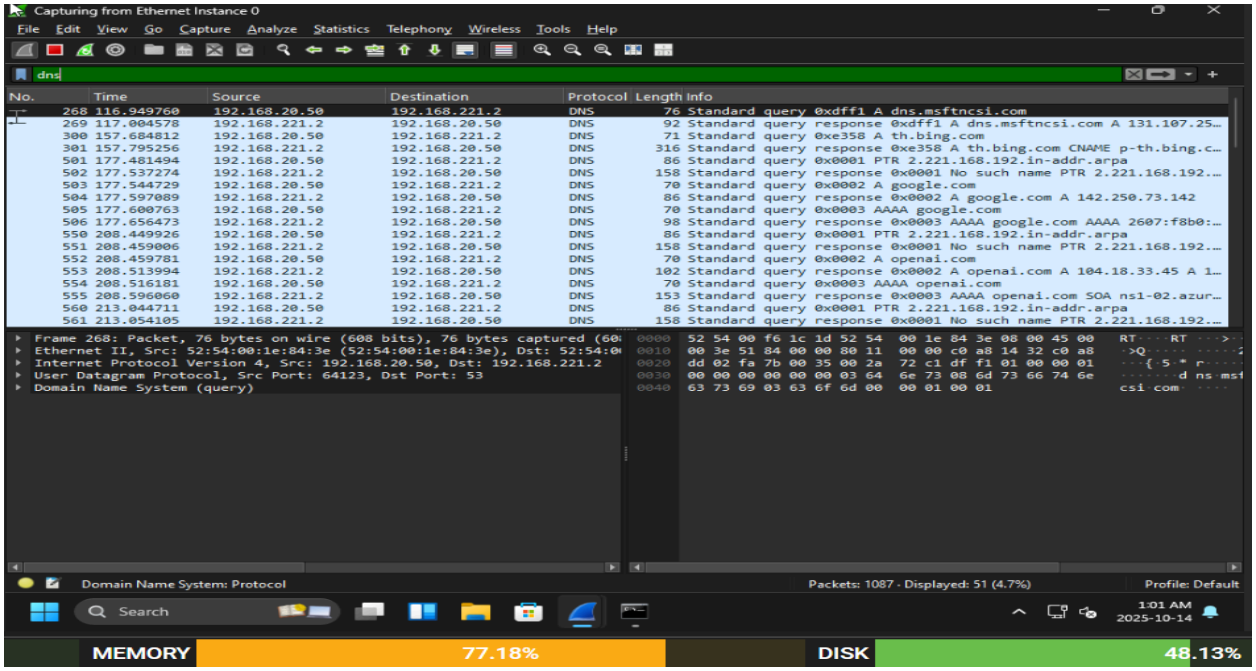


Fig13_DNS_Source

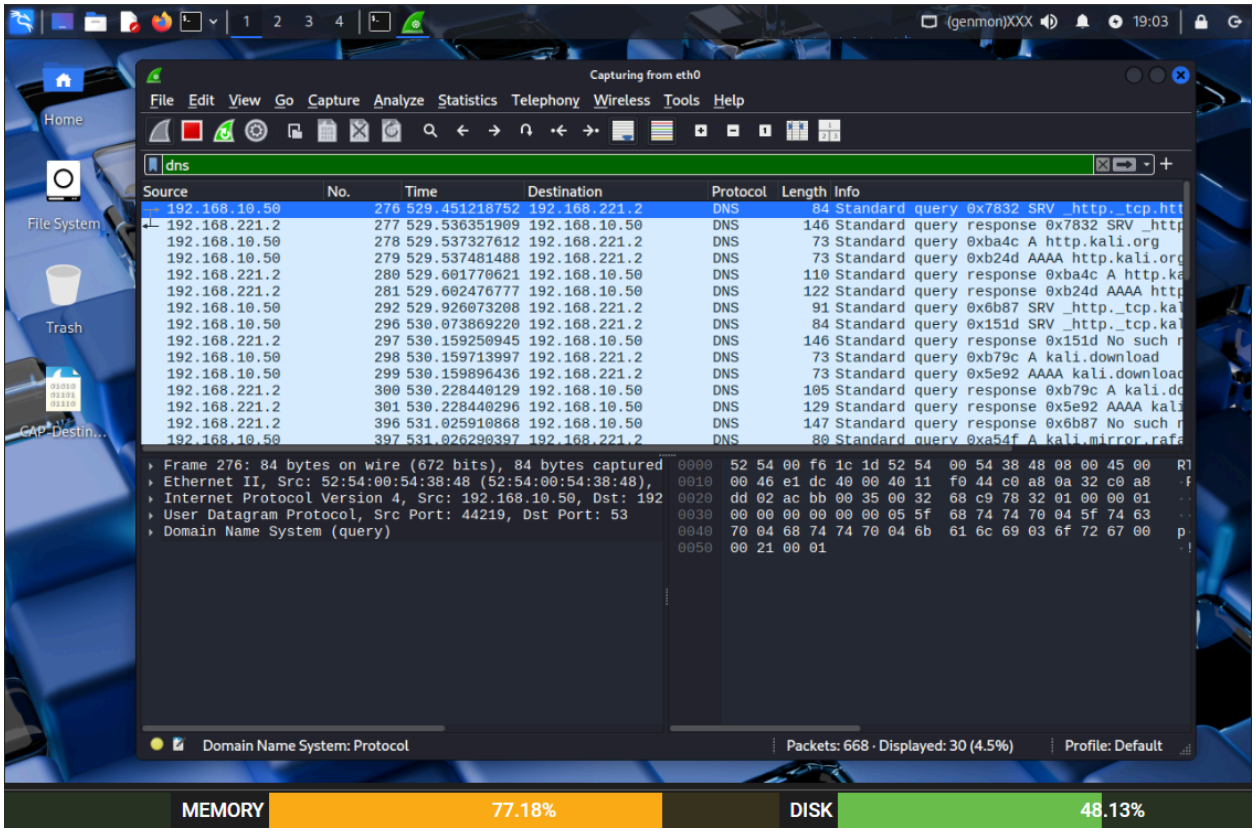


Fig14_DNS_Destination

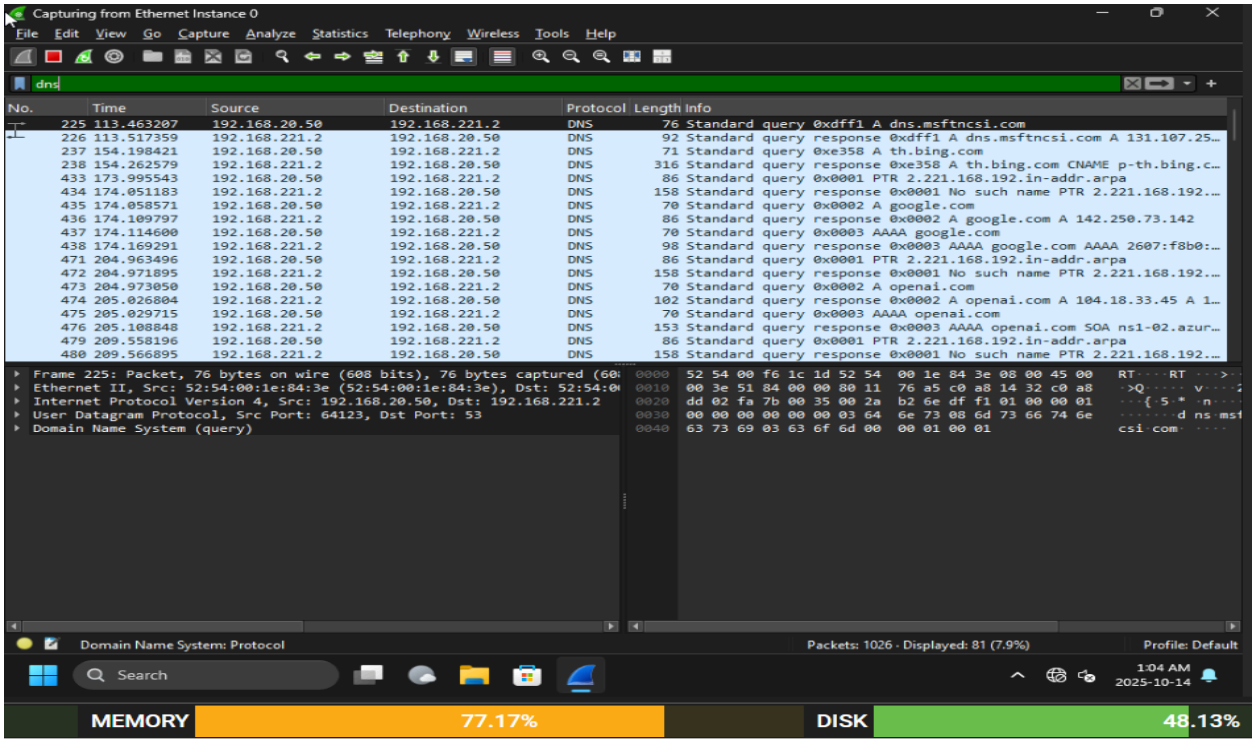


Fig15_DNS_SPAN

Explanation:

Each query uses **UDP** port 53. Queries and responses confirm **UDP** traffic flow across VLANs and **SPAN** mirroring.

Capture Files:
CAP-Source_Client20_DNS.pcapng
CAP-SPAN_Client30_DNS.pcapng
CAP-Destination_Kali_DNS.pcapng

9. HTTP Burst (Traffic Generation)

How this screenshot was captured:

From Client20: *(Powershell)*

for ($i=0; $i -lt 20; $i++) { Invoke-WebRequest -Uri http://192.168.10.50:8080 -UseBasicParsing }

Wireshark filter:

tcp.port==8080

Screenshot captured showing multiple HTTP GET requests in sequence.

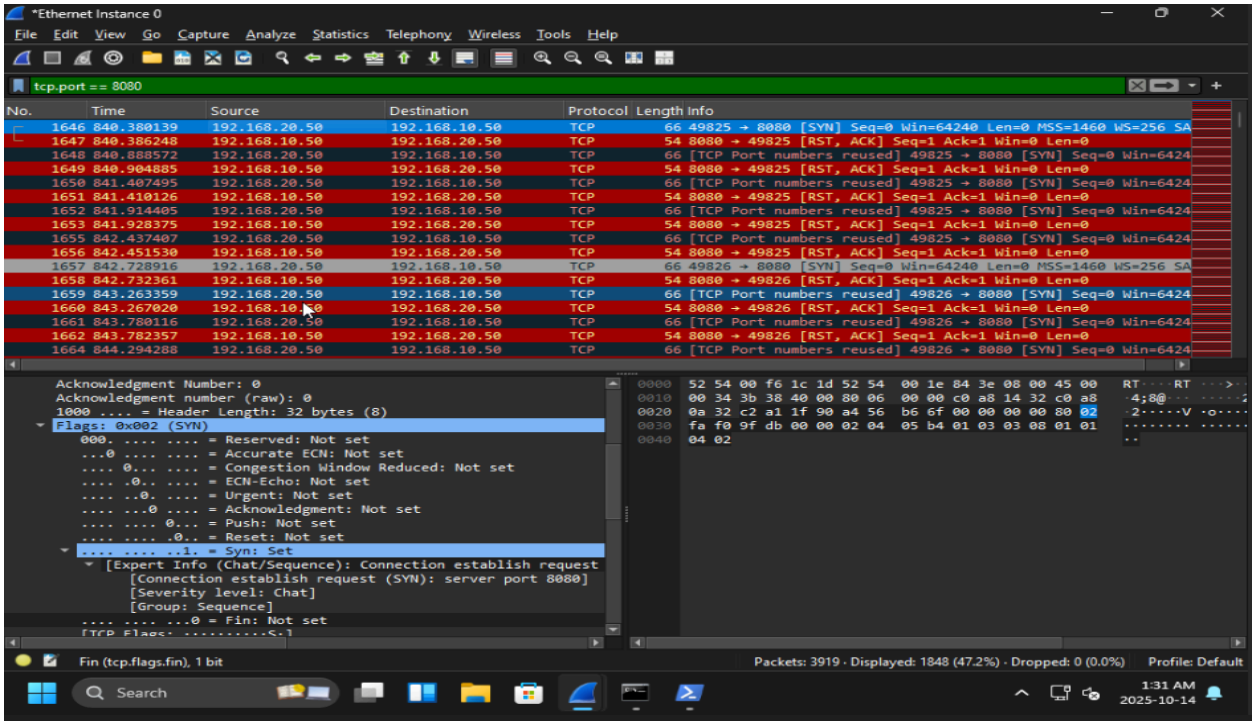


Fig16_HTTP_Burst

Explanation:

Multiple concurrent **TCP** sessions simulate real-world traffic and increase the total packet count to meet lab requirements.

Capture Files:
CAP-Source_Client20_HTTPBurst.pcapng
CAP-SPAN_Client30_HTTPBurst.pcapng
CAP-Destination_Kali_HTTPBurst.pcapng

10. TTL and MAC Address Analysis

How this screenshot was captured:

Display filter applied:

icmp && ip.src==192.168.20.50 && ip.dst==192.168.10.50

Expanded Ethernet II and IPv4 Header in the same packet across all captures to display TTL, source MAC, and destination MAC.

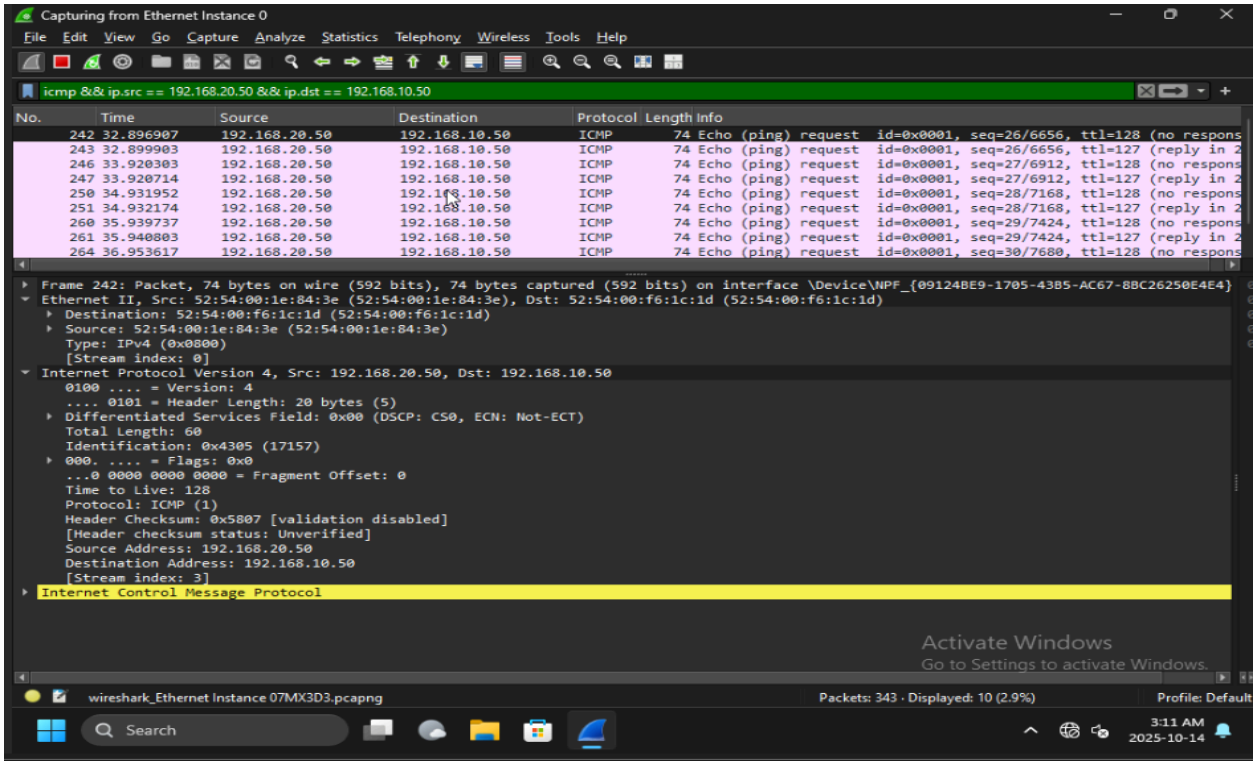


Fig17_TTL_Source

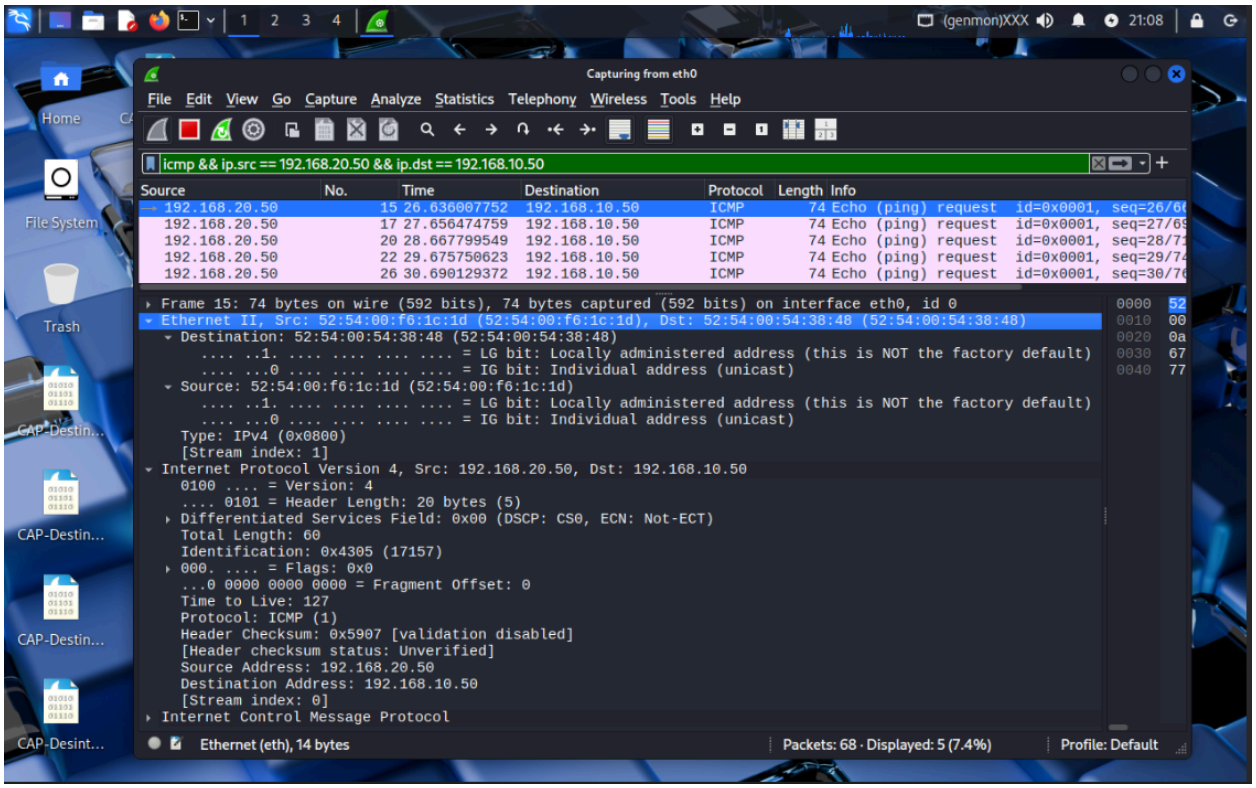


Fig18_TTL_Destination

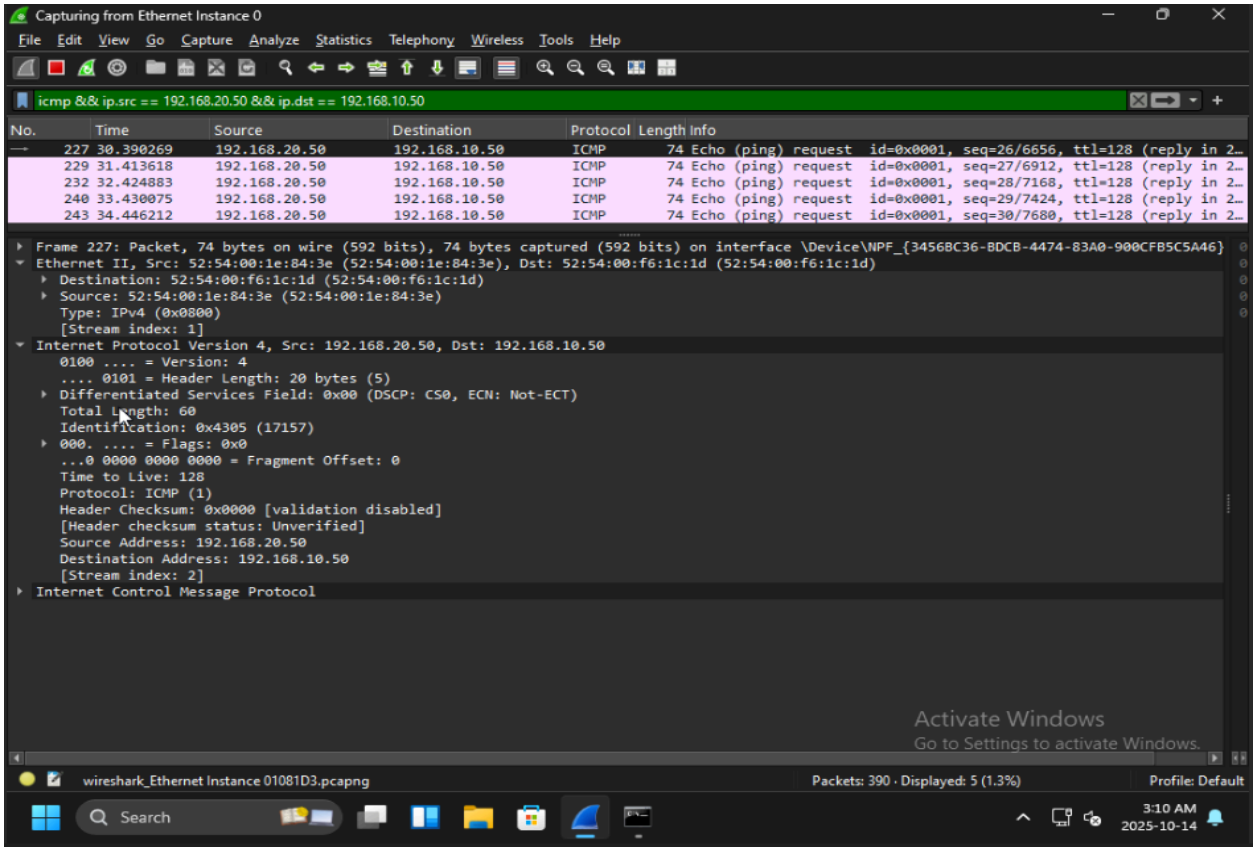


Fig19_TTL_SPAN

Explanation:

TTL decreases by one at each Layer 3 hop, and MAC addresses change at each Layer 2 segment since routers rewrite Ethernet headers before forwarding.

Capture Point	Filter Used	TTL	Source MAC	Destination MAC
Source (Client20)	same	[TTL value]	[MAC]	[MAC]
Destination (Kali)	same	[TTL value]	[MAC]	[MAC]
SPAN (Client30)	same	[TTL value]	[MAC]	[MAC]

Capture Files:
CAP-Source_Client20_TTL.pcapng
CAP-SPAN_Client30_TTL.pcapng
CAP-Destination_Kali_TTL.pcapng

11. Packet Count Verification

How this screenshot was captured:

In Wireshark:

Statistics → Capture File Properties

Each file shows a total packet count exceeding 3,000.

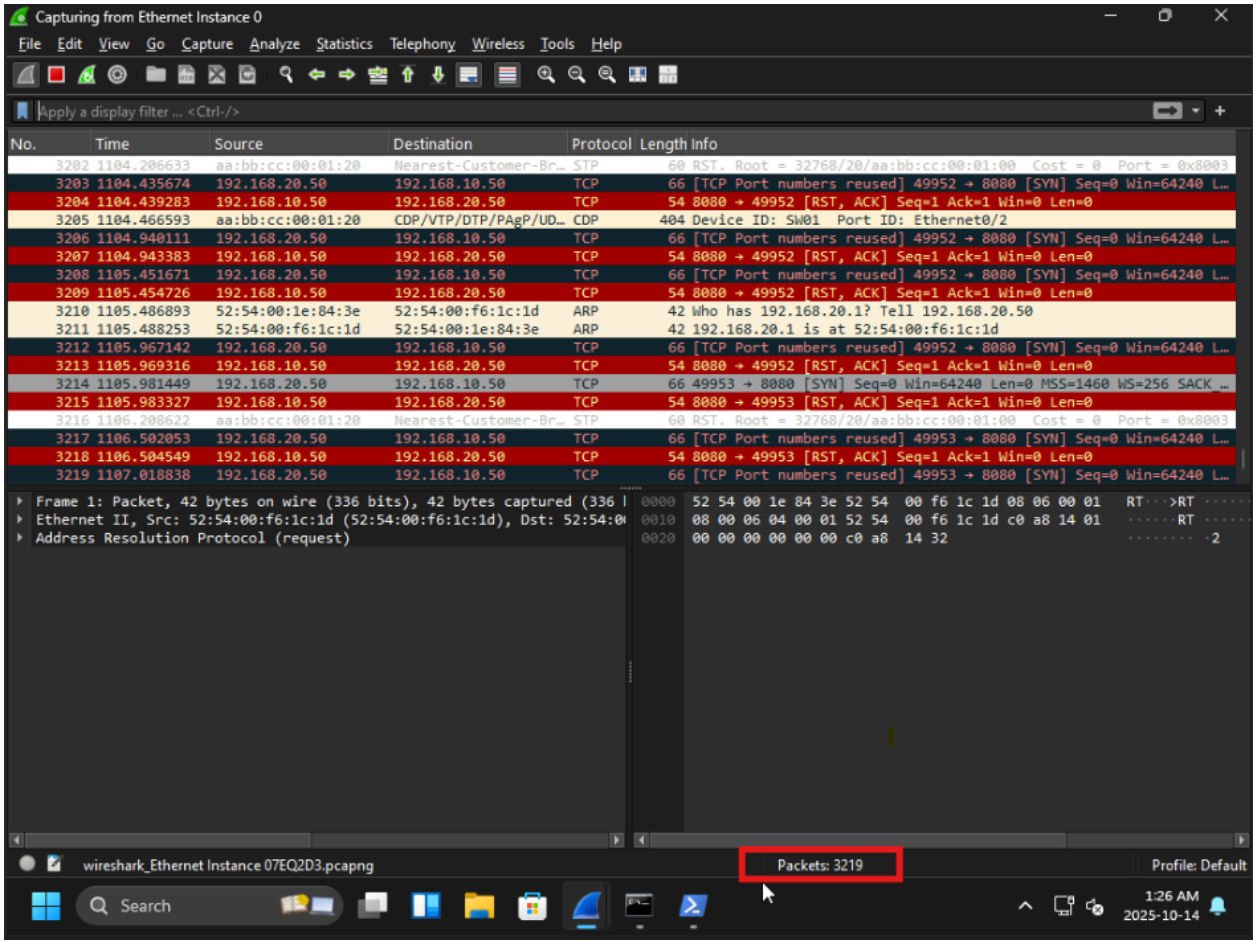


Fig20_TotalPackets

12. Conclusion

The Traffic Capture Lab demonstrated the ability to configure a **SPAN** session, generate and analyze traffic across VLANs, and use Wireshark to interpret **ICMP, TCP,** and **UDP** traffic. The **TCP three-way handshake** and **four-way teardown** were successfully identified, **DNS** verified **UDP** communication, and **TTL** analysis confirmed proper Layer 2/3 operation.

13. Submission Checklist

Submit as:
CYBR3010_TrafficCapture_GregoryStephens.zip

Contents:

- CYBR3010_TrafficCapture_Report_GregoryStephens.pdf
- All figures (Fig00 → Fig18)
- All .pcapng capture files:

CAP-Source_Client20_ICMP.pcapng
CAP-SPAN_Client30_ICMP.pcapng
CAP-Destination_Kali_ICMP.pcapng
CAP-Source_Client20_TCP8080.pcapng
CAP-SPAN_Client30_TCP8080.pcapng
CAP-Destination_Kali_TCP8080.pcapng
CAP-Source_Client20_DNS.pcapng
CAP-SPAN_Client30_DNS.pcapng
CAP-Destination_Kali_DNS.pcapng
CAP-Source_Client20_HTTPBurst.pcapng
CAP-SPAN_Client30_HTTPBurst.pcapng
CAP-Destination_Kali_HTTPBurst.pcapng
CAP-Source_Client20_TTL.pcapng
CAP-SPAN_Client30_TTL.pcapng
CAP-Destination_Kali_TTL.pcapng