Project Report

On

Network Traffic Analysis Using Wireshark and Zeek

By

Ishita Chaurasiya

Under the
Supervision of
Abhishek Pandey (IBM)

LLOYD INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA

AFFILIATED To

Dr. A P J ABDUL KALAM TECHNICAL

UNIVERSITY, LUCKNOW

September 2025

Abstract

This project demonstrates how to capture, analyze, and interpret network traffic using **Wireshark** (packet-level analysis) and **Zeek** (network security monitoring and structured logs). The report covers lab setup, ARP spoofing (MITM), capturing protocol-level plaintext leaks (HTTP, Telnet, SMTP), DHCP/DNS analysis, and how to correlate Zeek logs with Wireshark packet captures. The goal is to illustrate vulnerabilities in unencrypted protocols, compare manual packet inspection vs automated log analysis, and propose mitigation strategies.

Table of Contents

---

1. Introduction

Network traffic analysis is essential to understand how data moves across networks and to detect security weaknesses. Packet-level tools (Wireshark) let investigators inspect exact bytes on the wire, while network monitoring platforms (Zeek) convert traffic into structured logs that are easier to query and ingest into analytics systems.

This project replicates common insecure scenarios (ARP spoofing, unencrypted login, SMTP without STARTTLS, DHCP vulnerabilities) in a controlled lab and demonstrates detection and analysis using both Wireshark and Zeek.

---

2. Objectives

- Capture live network traffic in a controlled environment.
- Demonstrate ARP spoofing (Man-in-the-Middle) and inspect resulting traffic.
- Capture and identify plaintext credentials from HTTP and Telnet.
- Observe SMTP traffic before and after STARTTLS negotiation.
- Capture DHCP DORA process and explain its vulnerabilities.
- Use Zeek to produce structured logs and show how it helps detect the same events.
- Produce recommendations for securing the network.

---

3. Tools and Technologies

- **Wireshark** - packet capture and GUI analysis (https://www.wireshark.org)
- **Zeek** - network security monitor that produces logs (https://zeek.org)
- **Linux (Ubuntu / Kali)** - attacker machine for ARP spoofing and running tools
- **arpspoof / ettercap / Bettercap** – tools to perform ARP poisoning
- **tcpdump** – command-line packet capture (optional for large captures)
- **VirtualBox / VMware** – to create isolated lab VMs
- **Optional:** ELK stack (Elasticsearch, Logstash, Kibana) or Brim/Zeek-Cluster for log visualization

---

4. Lab Topology and Environment

Topology: - Victim VM (Windows or Linux) — 192.168.56.101 - Router / Gateway (Virtual or physical) — 192.168.56.1 - Attacker VM (Ubuntu/Kali) — 192.168.56.102 (runs Wireshark & Zeek)

Network: All VMs are on the same host-only or NAT network so they can communicate without affecting production networks. Use snapshots so you can revert any changes.

Precautions: - Perform experiments on an isolated test network or sandbox to avoid impacting real users. - Use intentionally vulnerable testing sites (e.g., VulnWeb, Damn Vulnerable Web Application) for HTTP tests.

---

5. Methodology (Step-by-step)

5.1 Prepare VMs and Tools

1. Create three VMs: Victim, Router/Internet (can be simulated by host), Attacker.
2. Install Wireshark and Zeek on the attacker VM. Install arpspoof/ettercap for MITM.

5.2 ARP Spoofing (MITM)

1. Enable IP forwarding on attacker:

```
# Enable IP forwarding (temporary)
sudo sysctl -w net.ipv4.ip_forward=1

# To persist, edit /etc/sysctl.conf and set net.ipv4.ip_forward=1
```

1. Launch arpspoof (dsniff) or bettercap:

```
# Using arpspoof (dsniff package)
sudo arpspoof -i eth0 -t <Victim_IP>
<Router_IP> sudo arpspoof -i eth0 -t
<Router_IP> <Victim_IP>

# Using bettercap (modern alternative)
sudo bettercap -iface eth0 -eval "set arp.spoof.targets <Victim_IP>;
arp.spoof on"
```

1. Start Wireshark on the attacker's interface (`eth0`) to capture traffic.

5.3 Simulate Victim Activities

- Open vulnerable test site and log in (HTTP) to capture credentials.
- Use `telnet victim-server 23` or another service on victim to generate Telnet traffic.
- Send an SMTP session via `telnet smtp.example.com 587` (observe STARTTLS negotiation).
- Run `ipconfig /release` and `ipconfig /renew` to capture DHCP exchange.

5.4 Capture and Save PCAPs

- In Wireshark: `File -> Save As` capture.pcap
- Or use `tcpdump` :

```
sudo tcpdump -i eth0 -w capture.pcap
```

---

6. Wireshark: Capture & Analysis

6.1 Applying Filters

- ARP: `arp`
- HTTP: `http` or `tcp.port == 80`
- Telnet: `telnet` or `tcp.port == 23`
- SMTP: `smtp` or `tcp.port == 25 || tcp.port == 587 || tcp.port == 465`
- DHCP: `bootp` (DHCP uses BOOTP dissector)

6.2 Inspecting Packets

- Select a packet and expand protocol layers (Ethernet -> IP -> TCP -> HTTP).
- For HTTP POSTs, inspect the `Hypertext Transfer` section to see form fields and credentials if unencrypted.
- For Telnet, inspect the payload bytes (Telnet is plain text).
- For SMTP on port 587, look for `STARTTLS` command; traffic before TLS handshake is plaintext.

6.3 Examples (what to look for)

- **ARP spoofing evidence:** Repeated ARP replies, wrong MAC-to-IP mapping in ARP replies.
- **Credentials:** HTTP POST body with `username` and `password` fields.
- **SMTP:** `AUTH LOGIN` and base64-encoded credentials before TLS.
- **DHCP:** DORA packets: Discover, Offer, Request, Ack – check offered gateway and DNS IPs.

---

7. Zeek: Installation, Running, and Log Analysis

7.1 Installing Zeek (Ubuntu)

```
# Add Zeek repository (example)
sudo apt update
sudo apt install -y zeek

# Alternatively, follow official build-from-source for latest version
```

7.2 Running Zeek on Live Interface

```
sudo zeek -i eth0 local
# or simply start zeek in manager mode if installed via packages
sudo zeekctl deploy
```

7.3 Processing PCAPs with Zeek

```
zeek -r capture.pcap
```

This generates structured logs in the current directory: `conn.log`, `http.log`, `dns.log`, `ssl.log`, `smtp.log`, and others.

7.4 Key Zeek Logs

- `conn.log` — connection metadata (start time, duration, protocol, bytes, state). Useful to find suspicious flows.

- `http.log` — HTTP requests and responses, URIs, user-agent; Zeek extracts fields rather than raw bytes.
- `smtp.log` — SMTP sessions, commands observed (but may not show full message bodies if encrypted).
- `dns.log` — DNS queries and responses.
- `dhcp.log` — DHCP transactions (if the DHCP analyzer is enabled).

7.5 Using Zeek Scripts for Detection

- Zeek ships with many scripts (policy scripts) to detect anomalies (e.g., `weird.log` for unusual events).
- You can write custom Zeek scripts to flag: plaintext credential patterns, repeated ARP anomalies, or suspicious DNS lookups.

Example simple script (detect HTTP POSTs containing login keywords):

```
# save as detect-login.zeek
@load base/protocols/http

event http_request(c: connection, method: string, original_URI: string,
unescaped_URI: string)
{
    if ( method == "POST" && /password|passwd|pwd/ in unescaped_URI )
        print fmt("Possible credential POST: %s %s", c$id$orig_h,
unescaped_URI);
}
```

Run Zeek loading script:

```
zeek -r capture.pcap detect-login.zeek
```

---

8. Correlating Wireshark & Zeek Findings

- Use timestamps and connection tuples (src IP, dst IP, src port, dst port) to map a Wireshark packet to a Zeek log entry.
- Example: A Wireshark packet with HTTP POST at timestamp `T` from `192.168.56.101:49152 -> 10.0.0.5:80` corresponds to a `http.log` entry in Zeek with identical tuple and timestamp range.
- Zeek helps reduce investigation time by surfacing suspicious flows; Wireshark provides byte-level evidence.

---

9. Results (Findings with example screenshots)

Note: Replace placeholder images with your own screenshots from your captures. Filenames are suggested and referenced in captions.

9.1 ARP Spoofing / MITM

- **Observation:** Attacker successfully poisoned ARP caches; victim traffic routed via attacker.
- **Evidence (Wireshark):** `arp` packets with forged MAC mappings and repeated gratuitous ARP replies.

Figure 1: ARP Spoofing capture showing forged ARP replies

- **Zeek evidence:** Look for unusual `weird.log` entries or unexpected traffic patterns from the victim to attacker.

9.2 HTTP Credential Capture

- **Observation:** HTTP POST requests reveal form data including username/password in plaintext.

Figure 2: Wireshark HTTP POST showing credentials in packet details

- **Zeek evidence:**`http.log` will show requested URI and often referer/user-agent. Zeek does not always log POST body by default, but can if scripted or if extracted via custom analyzers.

9.3 Telnet Plaintext Commands

- **Observation:** Telnet sessions show user commands and credentials unencrypted.

Figure 3: Telnet payload captured in Wireshark

- **Zeek evidence:** Telnet traffic appears in `conn.log` and `weird.log` (if configured). Extracting telnet payloads may require custom script or using `tcpdump` /Wireshark.

9.4 SMTP before STARTTLS

- **Observation:** SMTP session initiates in plaintext; if STARTTLS is not used, credentials or email content can be captured.

Figure 4: SMTP session capture showing AUTH command before TLS

- **Zeek evidence:** `smtp.log` may capture metadata. Watch for `AUTH` commands and absence of TLS handshake.

9.5 DHCP DORA

- **Observation:** DHCP Discover/Offer/Request/Ack broadcast messages; client accepts first offer and trusts server.

Figure 5: DHCP DORA exchange captured in Wireshark

- **Zeek evidence:** If DHCP parser enabled, check `dhcp.log` for the transaction and offered/assigned IPs.

---

10. Mitigation and Recommendations

- **Use encryption:** Replace Telnet with SSH, HTTP with HTTPS, and SMTP with SMTPS/STARTTLS.
- **Switch-level security:** Enable DHCP snooping, Dynamic ARP Inspection (DAI), and port security on managed switches.
- **Network segmentation:** Limit broadcast domains and restrict access to critical services.
- **Harden endpoints:** Ensure clients verify certificates, disable insecure protocols, and keep software up-to-date.
- **Monitoring & Alerting:** Deploy Zeek in production to produce logs feeding into SIEM/ELK and set alerts for suspicious events.
- **User awareness:** Train users not to use unsecured public Wi-Fi for sensitive activities.

---

11. Conclusion

This project showed how packet captures (Wireshark) and network logs (Zeek) complement each other: Wireshark provides definitive byte-level proof, while Zeek scales analysis to long-term monitoring and ingestion into analytics systems. The experiments reproduced typical vulnerabilities in unencrypted protocols and demonstrated concrete mitigations.

---

12. References

- Wireshark Documentation — https://www.wireshark.org/docs/
- Zeek — https://zeek.org/
- dsniff / arpspoof — https://www.monkey.org/~dugsong/dsniff/
- Bettercap — https://www.bettercap.org/

---

13. Appendix: Commands, Scripts, and Useful Tips

A. Useful Commands

```
# Enable IP forwarding
sudo sysctl -w net.ipv4.ip_forward=1

# Run arpspoof
sudo arpspoof -i eth0 -t 192.168.56.101 192.168.56.1
sudo arpspoof -i eth0 -t 192.168.56.1 192.168.56.101

# Capture with tcpdump
sudo tcpdump -i eth0 -w capture.pcap

# Run Zeek on capture
zeek -r capture.pcap

# Install Zeek on Debian/Ubuntu (package may
vary) sudo apt update && sudo apt install zeek -y

# Save Wireshark filtered view: use File -> Export Specified Packets
```

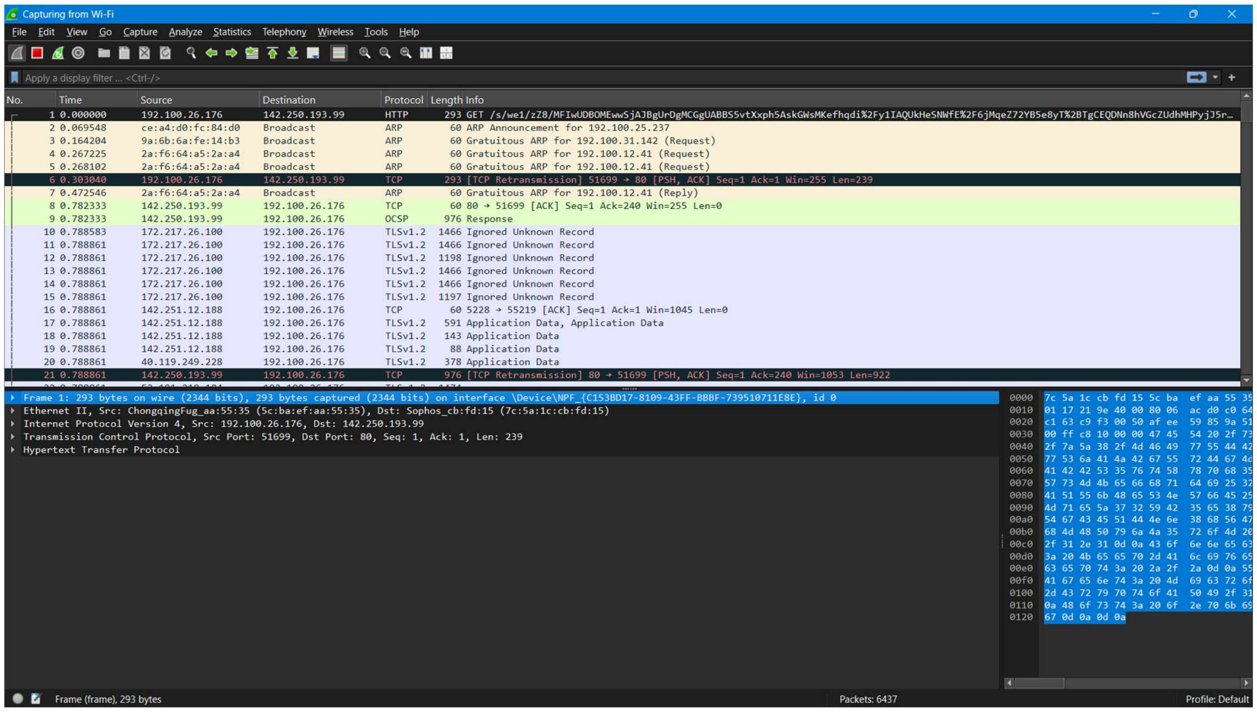
B. Example Zeek script to detect HTTP POST with password in URI

```
# detect-login.zeek
@load base/protocols/http

event http_request(c: connection, method: string, original_URI: string,
unescaped_URI: string)
{
    if ( method == "POST" && /password|passwd|pwd/ in unescaped_URI )
        print fmt("Possible credential POST: %s %s", c$id$orig_h,
unescaped_URI);
}
```

C. Tips for capturing clean logs

- Stop background updates and other traffic on VMs to reduce noise.
- Use capture filters (BPF) in tcpdump if you only want specific ports: `tcp port 80 or port 23 or port`
- Use timestamps consistently and sync VM clocks (e.g., `sudo ntpdate pool.ntp.org`).



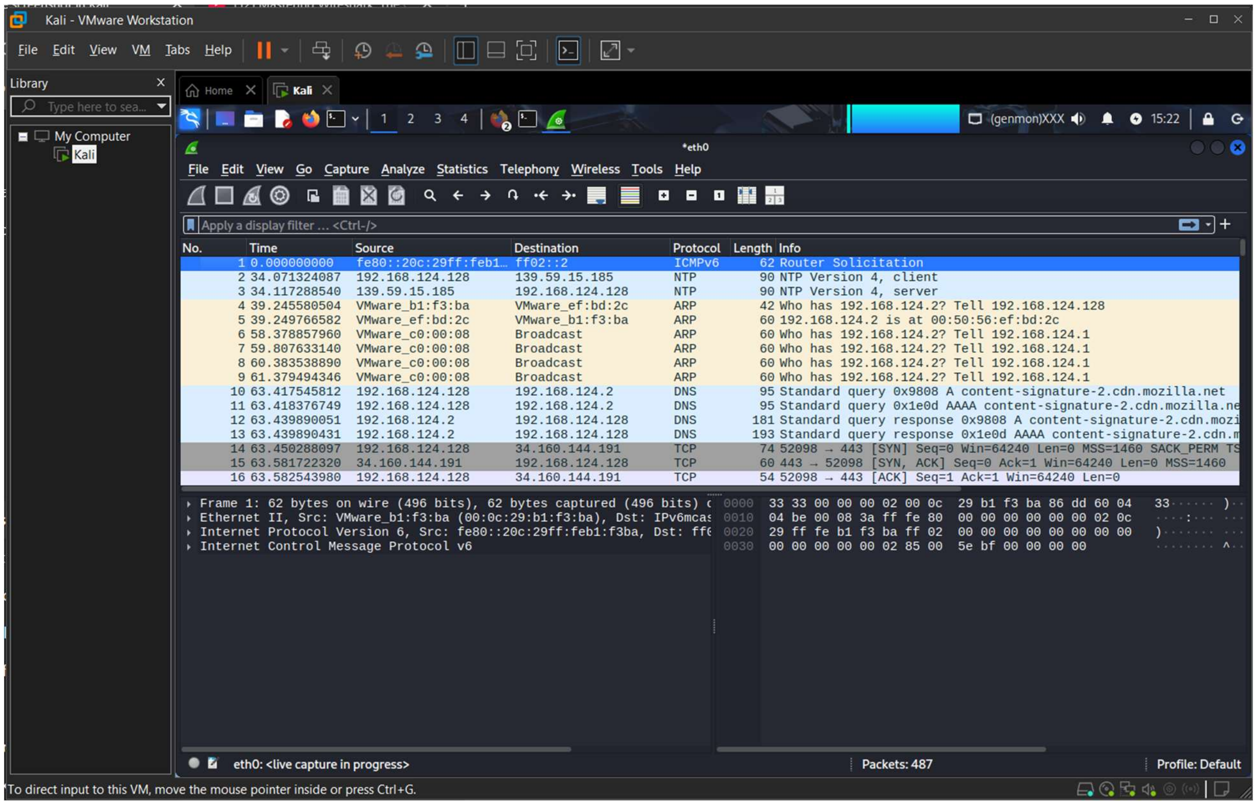
Capturing from Wi-Fi

```
No.  Time      Source             Destination      Protocol Length Info
   1 0.000000  192.100.26.176     142.250.193.99   HTTP      293 GET /s/we1/zZ8/MFIwUDBOMEwwSjAJBgUrDgMCGgUABBS5vtXxph5AskGWsMKefhqdi%2Fy1IAQUkHeSNWfE%2F6jMqeZ72YB5e8yT%2BTgCEQDNn8hVGcZUdhMHPyjJ5r…
   2 0.069548  ce:a4:d0:fc:84:d0  Broadcast        ARP        60 ARP Announcement for 192.100.25.237
   3 0.164204  9a:6b:6a:fe:14:b3  Broadcast        ARP        60 Gratuitous ARP for 192.100.31.142 (Request)
   4 0.267225  2a:f6:64:a5:2a:a4  Broadcast        ARP        60 Gratuitous ARP for 192.100.12.41 (Request)
   5 0.268102  2a:f6:64:a5:2a:a4  Broadcast        ARP        60 Gratuitous ARP for 192.100.12.41 (Request)
   6 0.303040  192.100.26.176     142.250.193.99   TCP       293 [TCP Retransmission] 51699 → 80 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=239
   7 0.472546  2a:f6:64:a5:2a:a4  Broadcast        ARP        60 Gratuitous ARP for 192.100.12.41 (Reply)
   8 0.782333  142.250.193.99     192.100.26.176   TCP        60 80 → 51699 [ACK] Seq=1 Ack=240 Win=255 Len=0
   9 0.782333  142.250.193.99     192.100.26.176   OCSP      976 Response
  10 0.788583  172.217.26.100     192.100.26.176   TLSv1.2  1466 Ignored Unknown Record
  11 0.788861  172.217.26.100     192.100.26.176   TLSv1.2  1466 Ignored Unknown Record
  12 0.788861  172.217.26.100     192.100.26.176   TLSv1.2  1198 Ignored Unknown Record
  13 0.788861  172.217.26.100     192.100.26.176   TLSv1.2  1466 Ignored Unknown Record
  14 0.788861  172.217.26.100     192.100.26.176   TLSv1.2  1466 Ignored Unknown Record
  15 0.788861  172.217.26.100     192.100.26.176   TLSv1.2  1197 Ignored Unknown Record
  16 0.788861  142.251.12.188     192.100.26.176   TCP        60 5228 → 55219 [ACK] Seq=1 Ack=1 Win=1045 Len=0
  17 0.788861  142.251.12.188     192.100.26.176   TLSv1.2   591 Application Data, Application Data
  18 0.788861  142.251.12.188     192.100.26.176   TLSv1.2   143 Application Data
  19 0.788861  142.251.12.188     192.100.26.176   TLSv1.2    88 Application Data
  20 0.788861  40.119.249.228     192.100.26.176   TLSv1.2   378 Application Data
  21 0.788861  142.250.193.99     192.100.26.176   TCP       976 [TCP Retransmission] 80 → 51699 [PSH, ACK] Seq=1 Ack=240 Win=1053 Len=922
```

```
› Frame 1: 293 bytes on wire (2344 bits), 293 bytes captured (2344 bits) on interface \Device\NPF_{C153BD17-8109-43FF-BBBF-739510711E8E}, id 0
› Ethernet II, Src: ChongqingFug_aa:55:35 (5c:ba:ef:aa:55:35), Dst: Sophos_cb:fd:15 (7c:5a:1c:cb:fd:15)
› Internet Protocol Version 4, Src: 192.100.26.176, Dst: 142.250.193.99
› Transmission Control Protocol, Src Port: 51699, Dst Port: 80, Seq: 1, Ack: 1, Len: 239
› Hypertext Transfer Protocol
```

```
0000  7c 5a 1c cb fd 15 5c ba  ef aa 55 35
0010  01 17 21 9e 40 00 80 06  ac d0 c0 64
0020  c1 63 c9 f3 00 50 af ee  59 85 9a 51
0030  00 ff c8 10 00 00 47 45  54 20 2f 73
0040  2f 7a 5a 38 2f 4d 46 49  77 55 44 42
0050  77 53 6a 41 4a 42 67 55  72 44 67 4c
0060  41 42 42 53 35 76 74 58  78 70 68 35
0070  57 73 4d 4b 65 66 68 71  64 69 25 32
0080  41 51 55 6b 48 65 53 4e  57 66 45 25
0090  4d 71 65 5a 37 32 59 42  35 65 38 79
00a0  54 67 43 45 51 44 4e 6e  38 68 56 47
00b0  68 4d 48 50 79 6a 4a 35  72 6f 4d 2e
00c0  2f 31 2e 31 0d 0a 43 6f  6e 6e 65 63
00d0  3a 20 4b 65 65 70 2d 41  6c 69 76 65
00e0  63 65 70 74 3a 20 2a 2f  2a 0d 0a 55
00f0  41 67 65 6e 74 3a 20 4d  69 63 72 6f
0100  2d 43 72 79 70 74 6f 41  50 49 2f 31
0110  0a 48 6f 73 74 3a 20 6f  2e 70 6b 69
0120  67 0d 0a 0d 0a
```

Frame (frame), 293 bytes   Packets: 6437   Profile: Default



Kali - VMware Workstation

Library: My Computer › Kali

*eth0

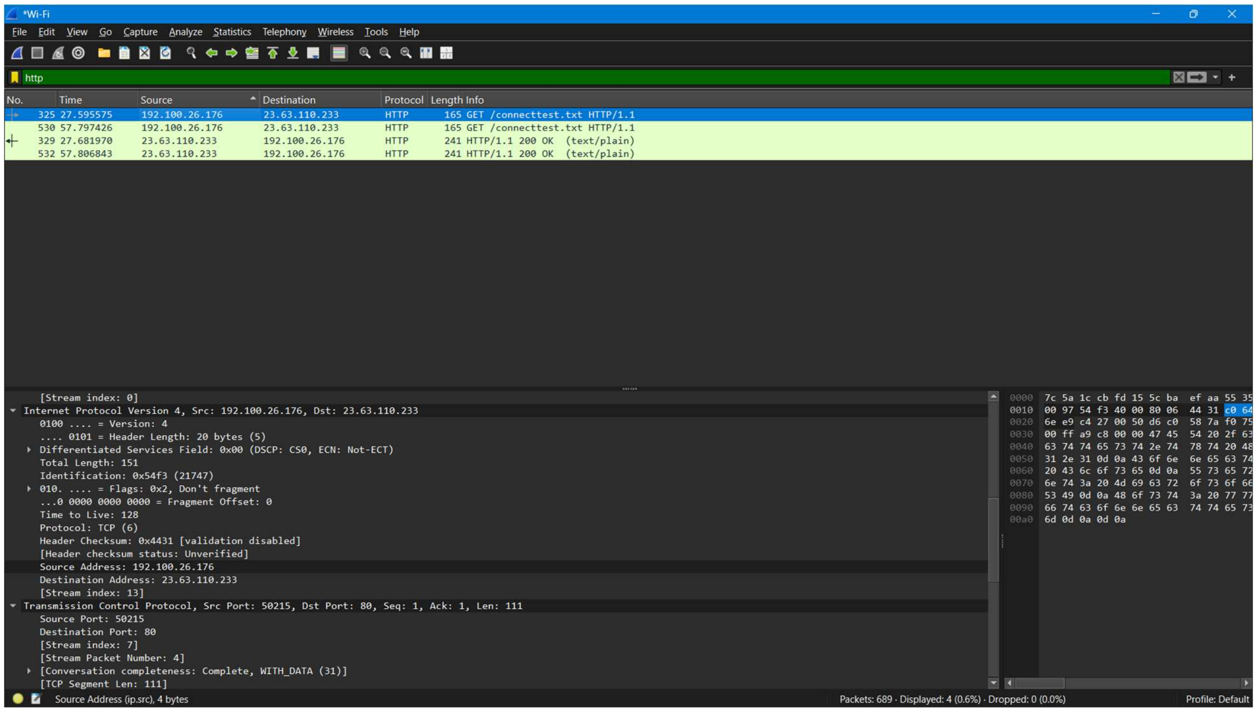
```
No.  Time          Source               Destination       Protocol Length Info
   1 0.000000000   fe80::20c:29ff:feb1… ff02::2           ICMPv6     62 Router Solicitation
   2 34.071324087  192.168.124.128      139.59.15.185     NTP        90 NTP Version 4, client
   3 34.117288540  139.59.15.185        192.168.124.128   NTP        90 NTP Version 4, server
   4 39.245580504  VMware_b1:f3:ba      VMware_ef:bd:2c   ARP        42 Who has 192.168.124.2? Tell 192.168.124.128
   5 39.249766582  VMware_ef:bd:2c      VMware_b1:f3:ba   ARP        60 192.168.124.2 is at 00:50:56:ef:bd:2c
   6 58.378857960  VMware_c0:00:08      Broadcast         ARP        60 Who has 192.168.124.2? Tell 192.168.124.1
   7 59.807633140  VMware_c0:00:08      Broadcast         ARP        60 Who has 192.168.124.2? Tell 192.168.124.1
   8 60.383538890  VMware_c0:00:08      Broadcast         ARP        60 Who has 192.168.124.2? Tell 192.168.124.1
   9 61.379494346  VMware_c0:00:08      Broadcast         ARP        60 Who has 192.168.124.2? Tell 192.168.124.1
  10 63.417545812  192.168.124.128      192.168.124.2     DNS        95 Standard query 0x9808 A content-signature-2.cdn.mozilla.net
  11 63.418376749  192.168.124.128      192.168.124.2     DNS        95 Standard query 0x1e0d AAAA content-signature-2.cdn.mozilla.ne
  12 63.439890051  192.168.124.2        192.168.124.128   DNS       181 Standard query response 0x9808 A content-signature-2.cdn.mozi
  13 63.439890431  192.168.124.2        192.168.124.128   DNS       193 Standard query response 0x1e0d AAAA content-signature-2.cdn.m
  14 63.450288097  192.168.124.128      34.160.144.191    TCP        74 52098 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TS
  15 63.581722320  34.160.144.191       192.168.124.128   TCP        60 443 → 52098 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  16 63.582543980  192.168.124.128      34.160.144.191    TCP        54 52098 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
```

```
› Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) c
› Ethernet II, Src: VMware_b1:f3:ba (00:0c:29:b1:f3:ba), Dst: IPv6mcas
› Internet Protocol Version 6, Src: fe80::20c:29ff:feb1:f3ba, Dst: ff0
› Internet Control Message Protocol v6
```

```
0000  33 33 00 00 00 02 00 0c  29 b1 f3 ba 86 dd 60 04   33······ )··
0010  04 be 00 08 3a ff fe 80  00 00 00 00 00 00 02 0c   ····:··· ···
0020  29 ff fe b1 f3 ba ff 02  00 00 00 00 00 00 00 00   )······· ···
0030  00 00 00 00 00 02 85 00  5e bf 00 00 00 00         ········ ^··
```

eth0: <live capture in progress>   Packets: 487   Profile: Default

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

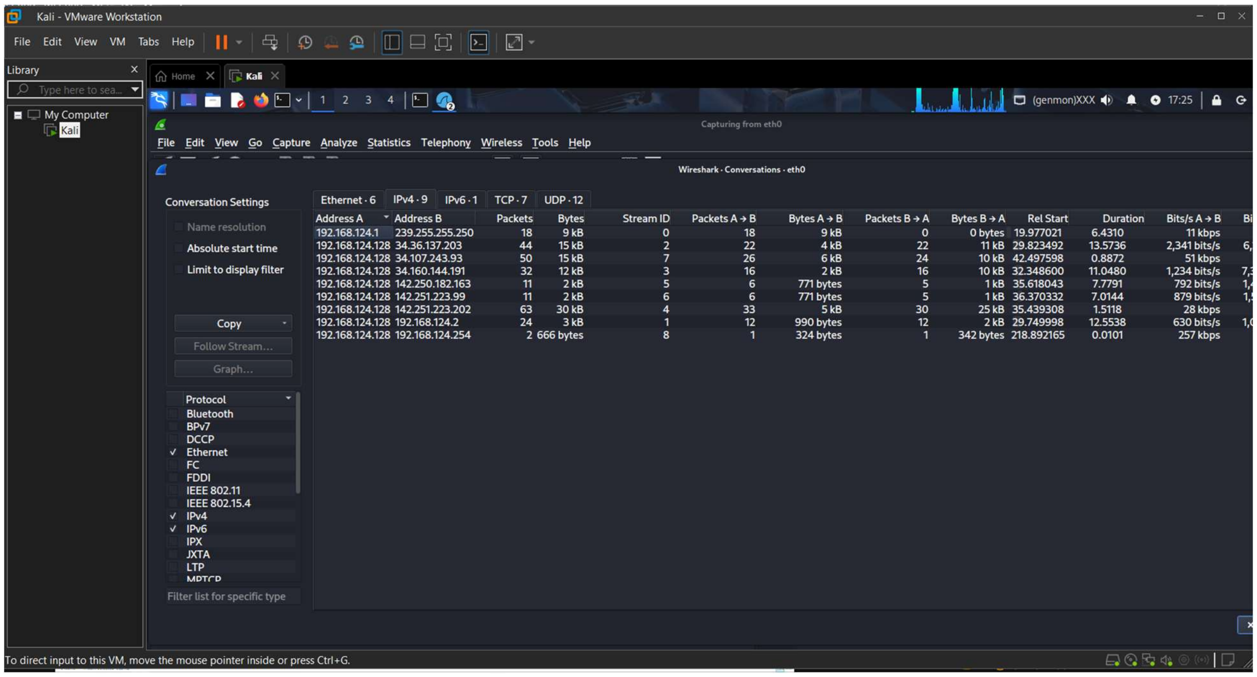

*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.   Time       Source           Destination      Protocol Length Info
  325 27.595575  192.100.26.176   23.63.110.233    HTTP        165 GET /connecttest.txt HTTP/1.1
  530 57.797426  192.100.26.176   23.63.110.233    HTTP        165 GET /connecttest.txt HTTP/1.1
  329 27.681970  23.63.110.233    192.100.26.176   HTTP        241 HTTP/1.1 200 OK  (text/plain)
  532 57.806843  23.63.110.233    192.100.26.176   HTTP        241 HTTP/1.1 200 OK  (text/plain)
```

```
    [Stream index: 0]
 Internet Protocol Version 4, Src: 192.100.26.176, Dst: 23.63.110.233
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 151
    Identification: 0x54f3 (21747)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x4431 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.100.26.176
    Destination Address: 23.63.110.233
    [Stream index: 13]
 Transmission Control Protocol, Src Port: 50215, Dst Port: 80, Seq: 1, Ack: 1, Len: 111
    Source Port: 50215
    Destination Port: 80
    [Stream index: 7]
    [Stream Packet Number: 4]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 111]
```

```
0000  7c 5a 1c cb fd 15 5c ba  ef aa 55 35
0010  00 97 54 f3 40 00 80 06  44 31 c0 64
0020  6e e9 c4 27 00 50 d6 c0  58 7a f0 75
0030  00 ff a9 c8 00 00 47 45  54 20 2f 63
0040  63 74 74 65 73 74 2e 74  78 74 20 48
0050  31 2e 31 0d 0a 43 6f 6e  6e 65 63 74
0060  20 43 6c 6f 73 65 0d 0a  55 73 65 72
0070  6e 74 3a 20 4d 69 63 72  6f 73 6f 66
0080  53 49 0d 0a 48 6f 73 74  3a 20 77 77
0090  66 74 63 6f 6e 6e 65 63  74 74 65 73
00a0  6d 0d 0a 0d 0a
```

Source Address (ip.src), 4 bytes   Packets: 689 · Displayed: 4 (0.6%) · Dropped: 0 (0.0%)   Profile: Default



Address A	Address B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bi
192.168.124.1	239.255.255.250	18	9 kB	0	18	9 kB	0	0 bytes	19.977021	6.4310	11 kbps	
192.168.124.128	34.36.137.203	44	15 kB	2	22	4 kB	22	11 kB	29.823492	13.5736	2,341 bits/s	6,
192.168.124.128	34.107.243.93	50	15 kB	7	26	6 kB	24	10 kB	42.497598	0.8872	51 kbps	
192.168.124.128	34.160.144.191	32	12 kB	3	16	2 kB	16	10 kB	32.348600	11.0480	1,234 bits/s	7,3
192.168.124.128	142.250.182.163	11	2 kB	5	6	771 bytes	5	1 kB	35.618043	7.7791	792 bits/s	1,4
192.168.124.128	142.251.223.99	11	2 kB	6	6	771 bytes	5	1 kB	36.370332	7.0144	879 bits/s	1,5
192.168.124.128	142.251.223.202	63	30 kB	4	33	5 kB	30	25 kB	35.439308	1.5118	28 kbps	
192.168.124.128	192.168.124.2	24	3 kB	1	12	990 bytes	12	2 kB	29.749998	12.5538	630 bits/s	1,0
192.168.124.128	192.168.124.254	2	666 bytes	8	1	324 bytes	1	342 bytes	218.892165	0.0101	257 kbps	