**SEMESTER PROJECT**

**INFORMATION SECURITY**

**[CLO-3]**



**Submitted By:**

Wabil Nadeem

Abdullah

Habiba Fiaz

Zahra Akhtar

Safa Ghaffar

**Session 2021-2025**

**University of Management and Technology**

# Table of Contents

[1. Tool Introduction 3](#_Toc200826694)

[2. Installation Steps (Screenshots) 5](#_Toc200826695)

[3. Tool Use Cases 8](#_Toc200826696)

[4. OSI Layer Mapping 21](#_Toc200826697)

[5. MITRE ATT&CK Mapping 22](#_Toc200826698)

[6. Mitigation Strategies 23](#_Toc200826699)

[7. References 24](#_Toc200826700)

# 1. Tool Introduction

Wireshark is a world-renowned open-source network protocol analyzer. It enables real-time traffic capture and deep inspection of network protocols across the OSI model. The tool is widely used in cybersecurity, network diagnostics, penetration testing, and academic research. Its primary strength lies in offering visibility into raw packets and headers for detailed analysis.

Key Features:

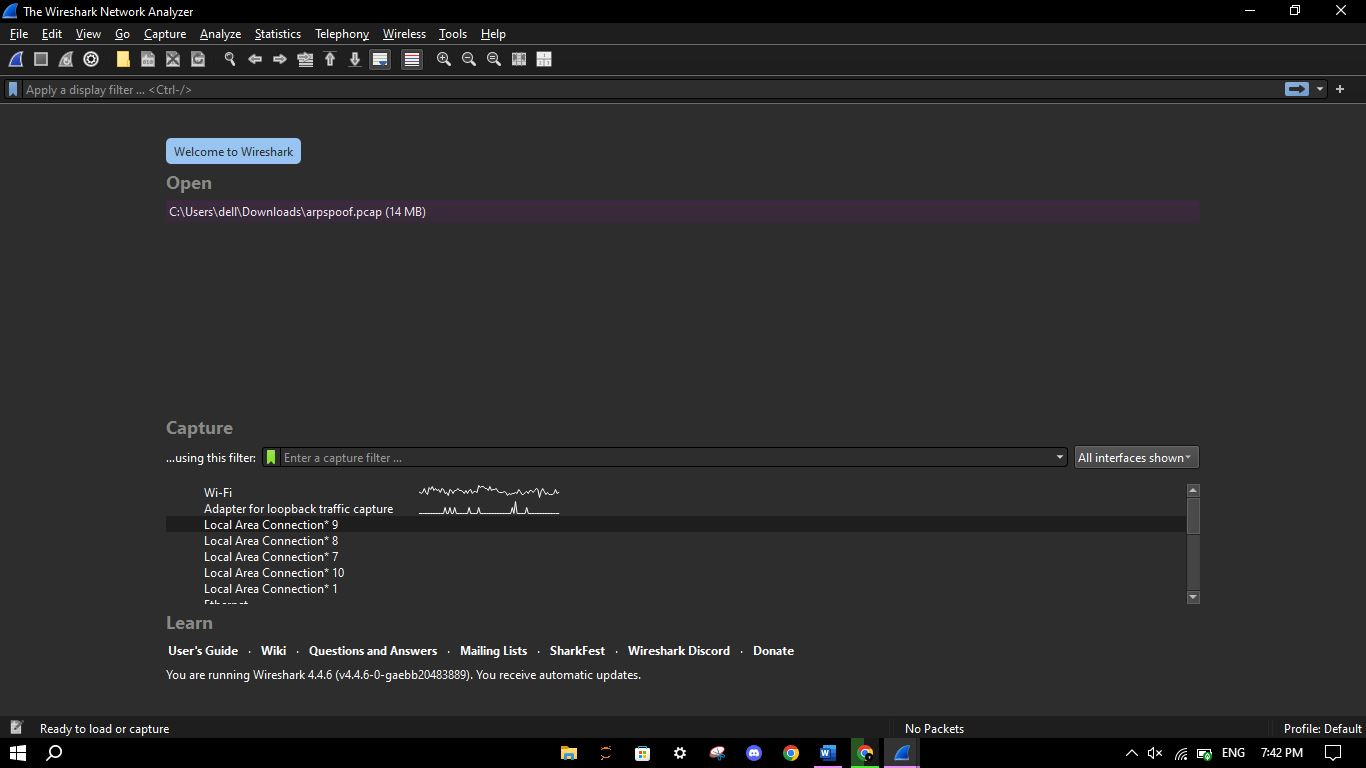
* - Real-time packet capture and filtering
* - Supports hundreds of network protocols
* - Interface selection and fine-grained filtering

# 2. Installation Steps (Screenshots)

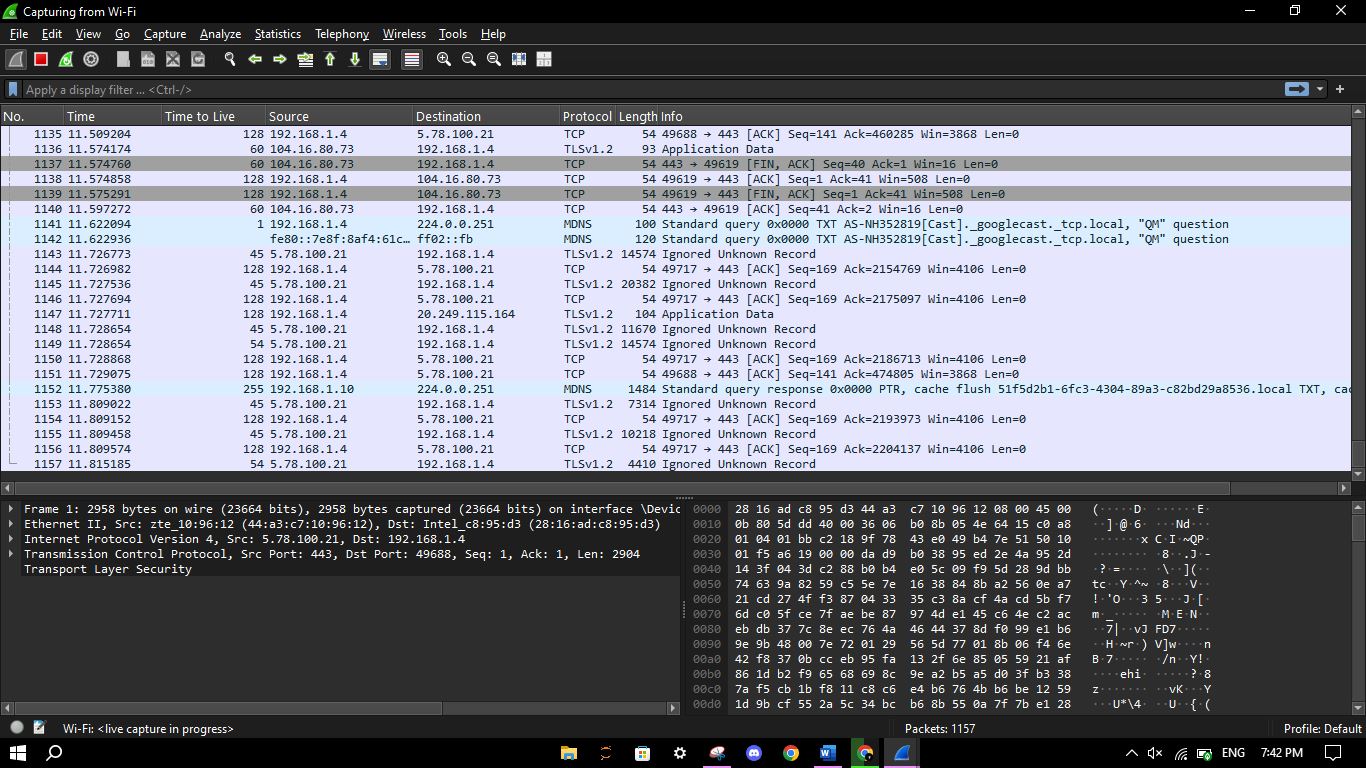
1. Visit https://www.wireshark.org and download the latest version.



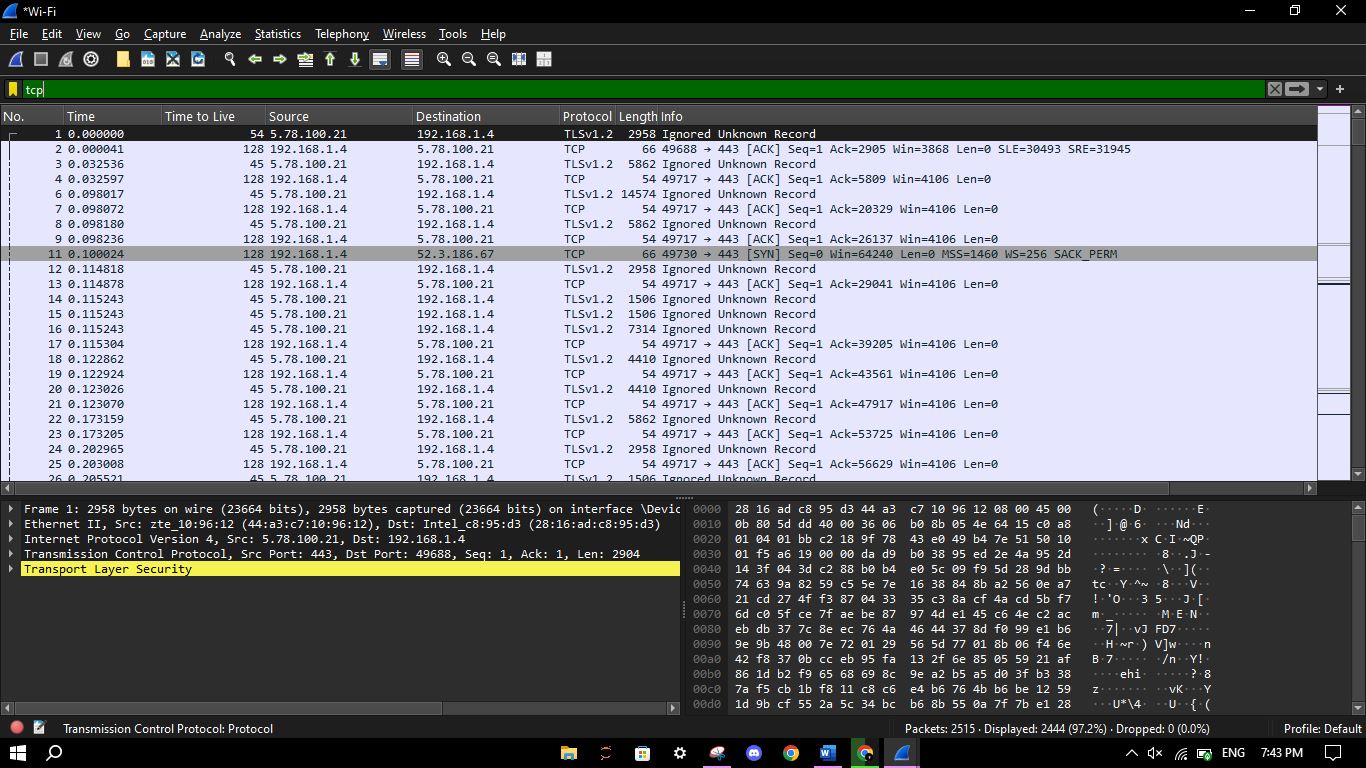
1. Launch Wireshark and select an interface (e.g., Wi-Fi, Loopback).

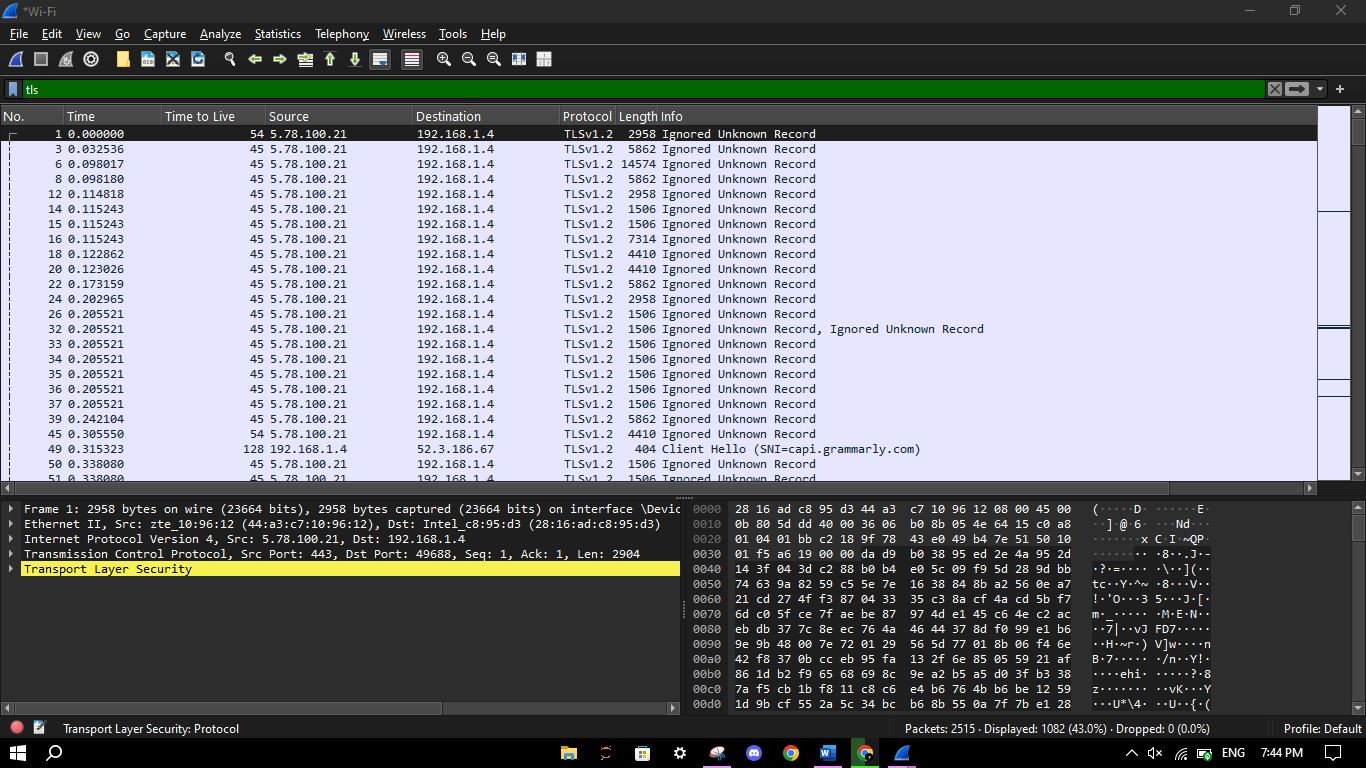


1. Click the blue shark fin icon to start capturing packets.



1. Use filters to focus on specific traffic (e.g., 'tcp', 'ip', 'http').





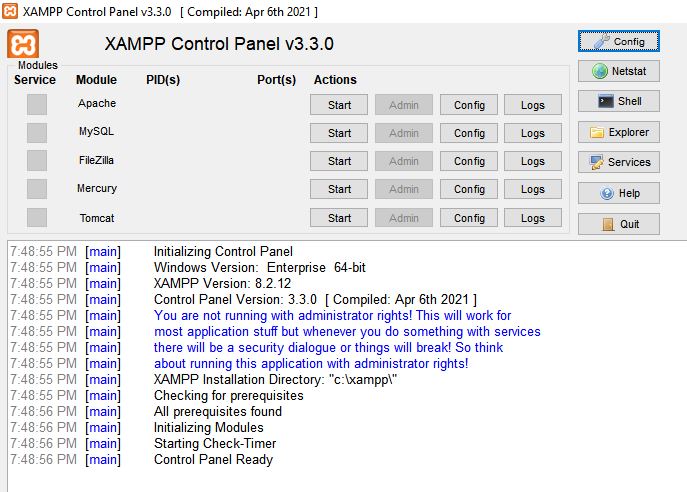
# 3. Tool Use Cases

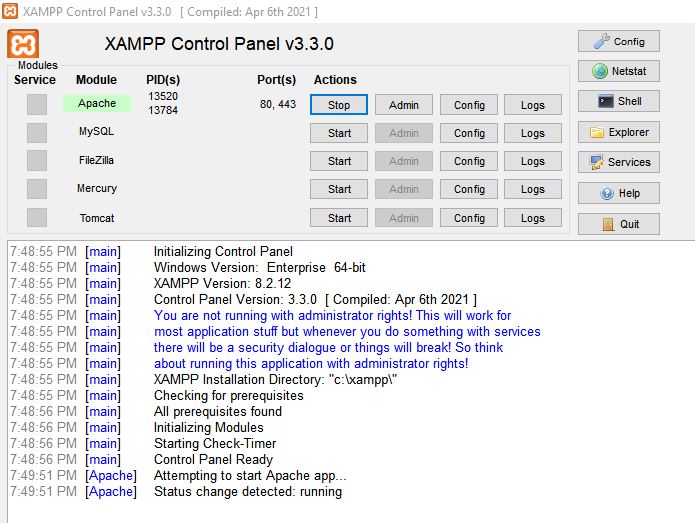
## Use Case 1: SYN Flood Attack Detection

Objective: Simulate and detect a SYN flood using spoofed IPs via Scapy.

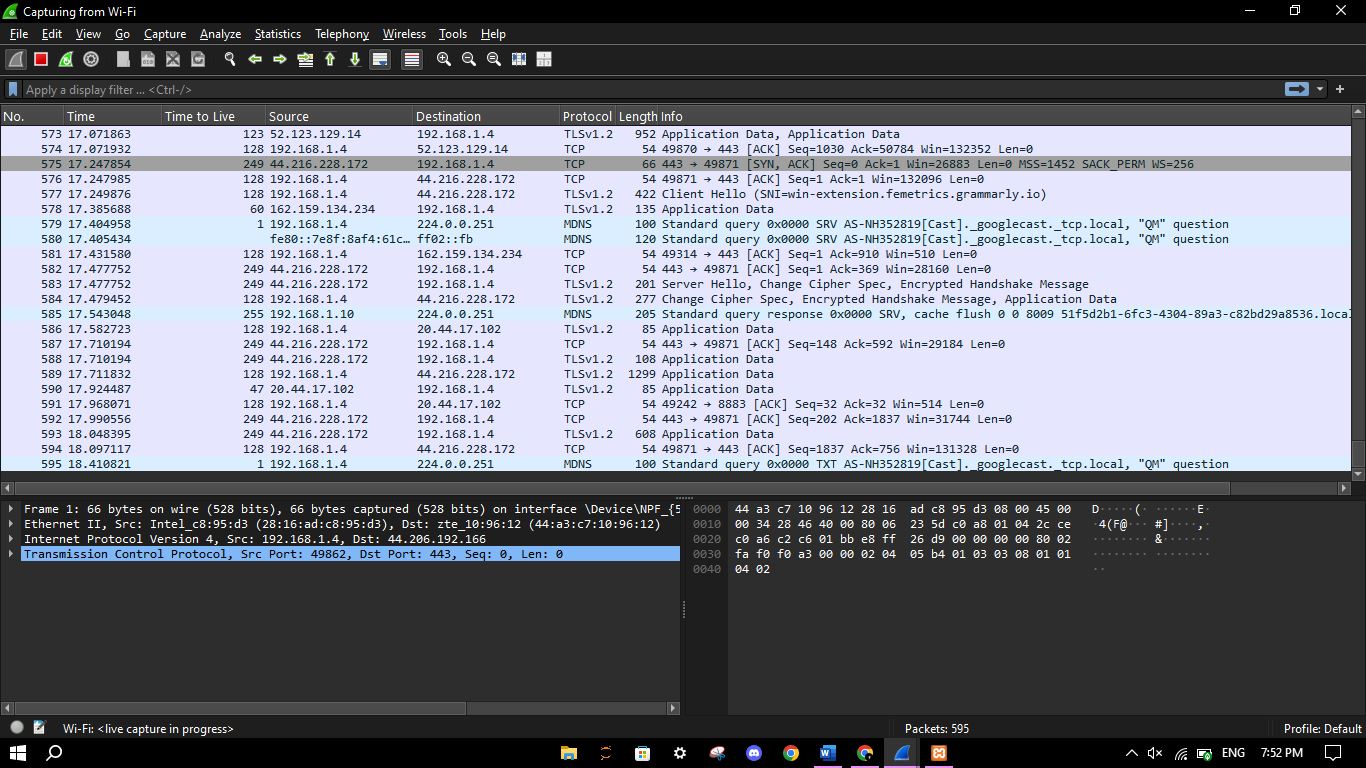
Process:

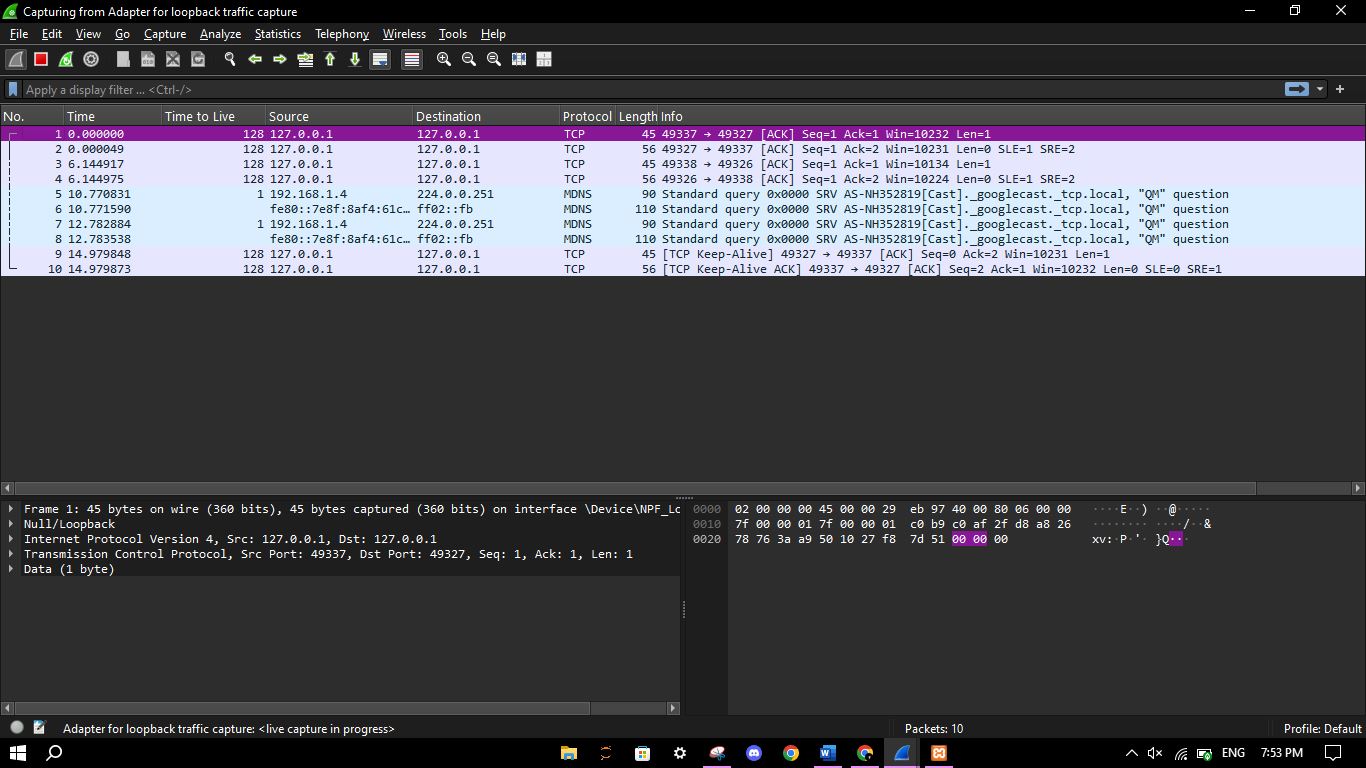
1. Host a website locally using XAMPP.



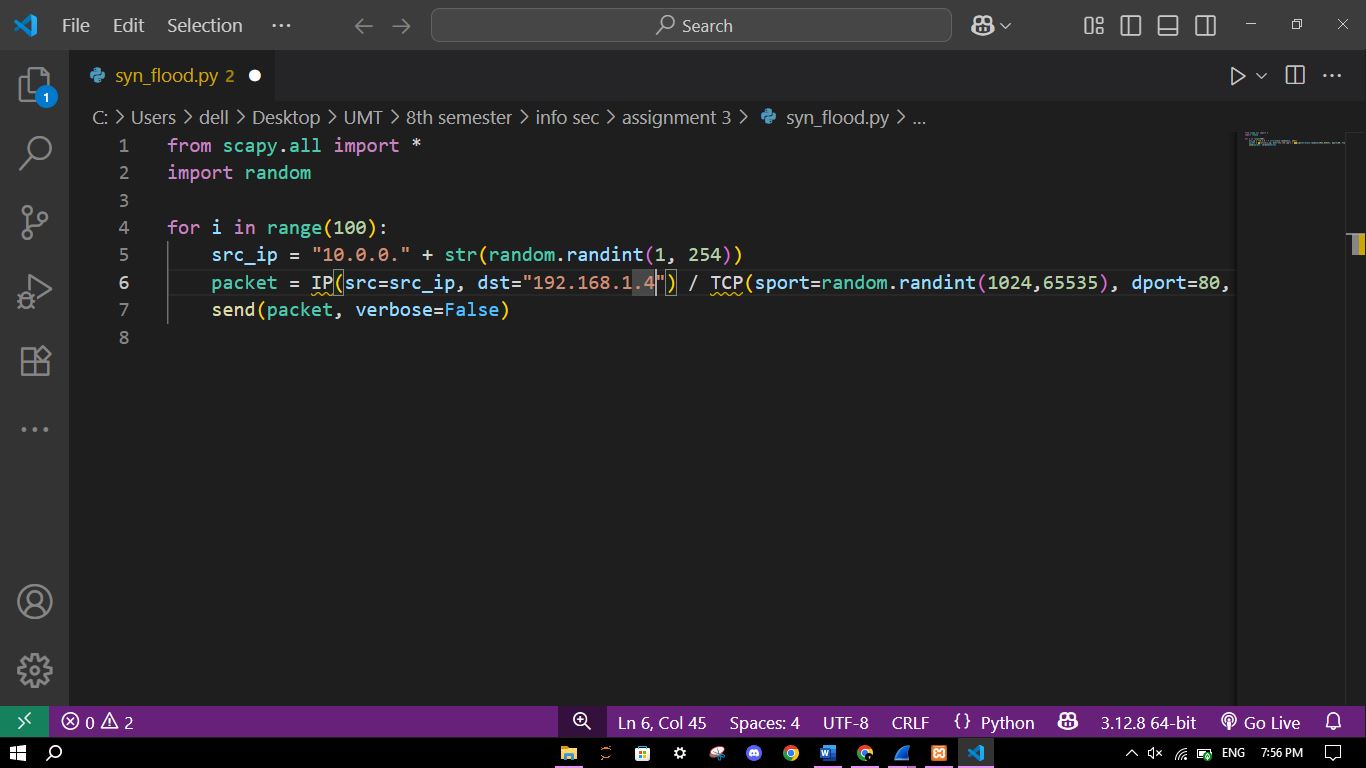


1. Start Wireshark on Loopback and Wi-Fi interfaces.

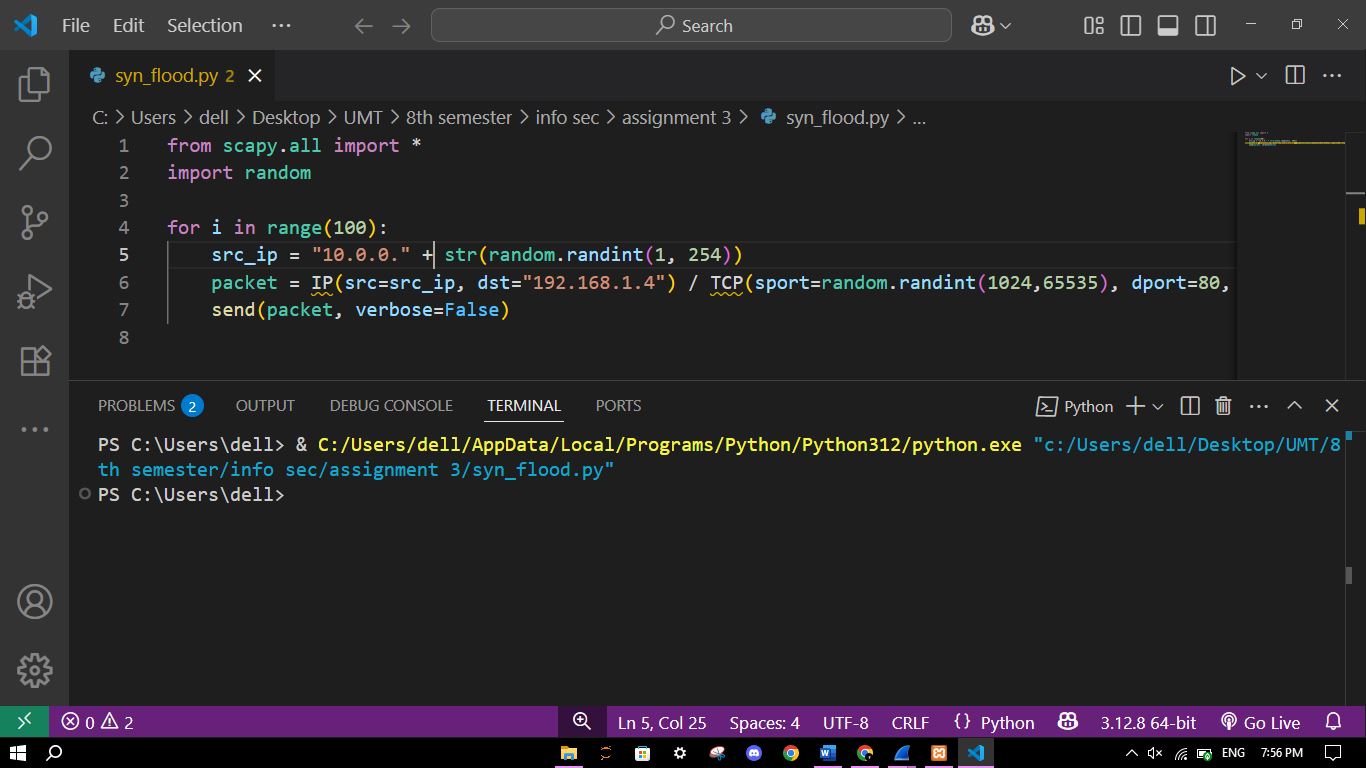




1. Run Scapy script to send spoofed SYN packets with random IPs.

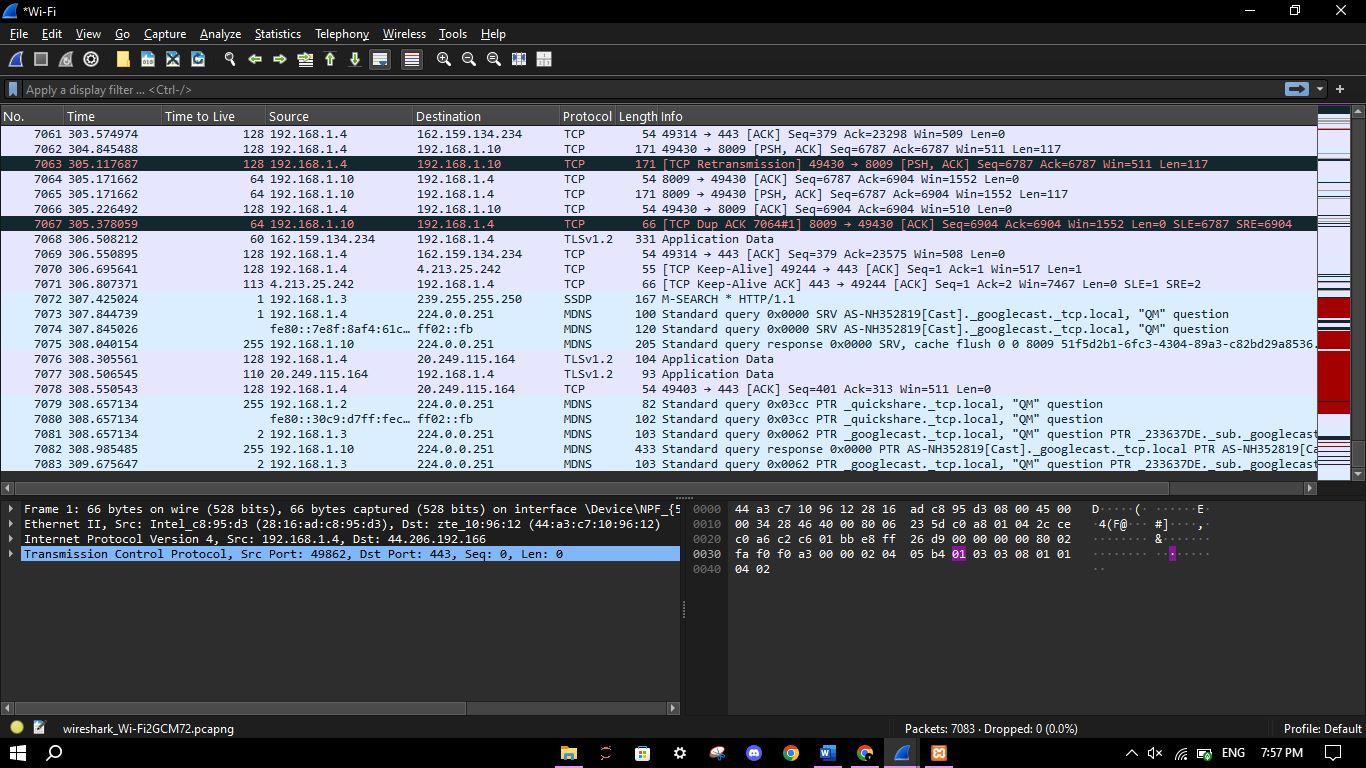


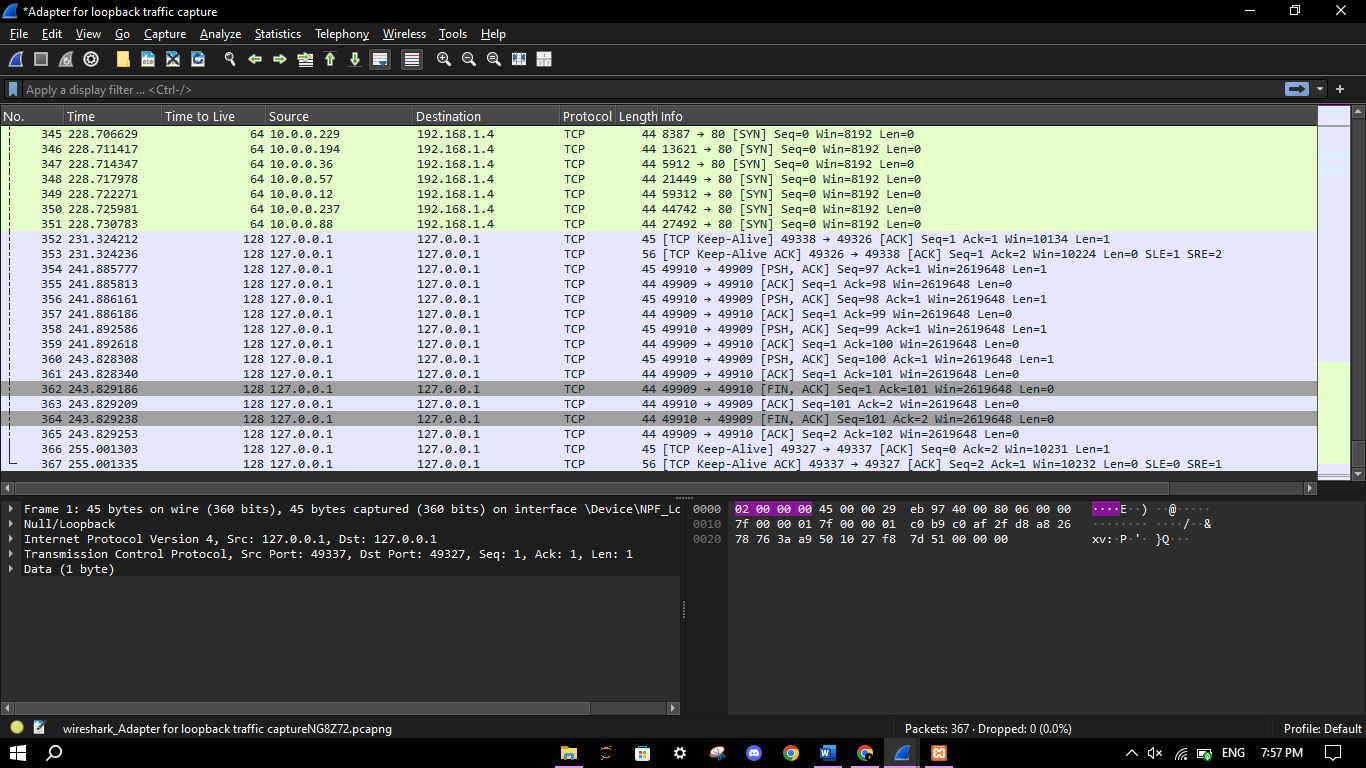






1. Use filters to analyze SYNs and ACKs

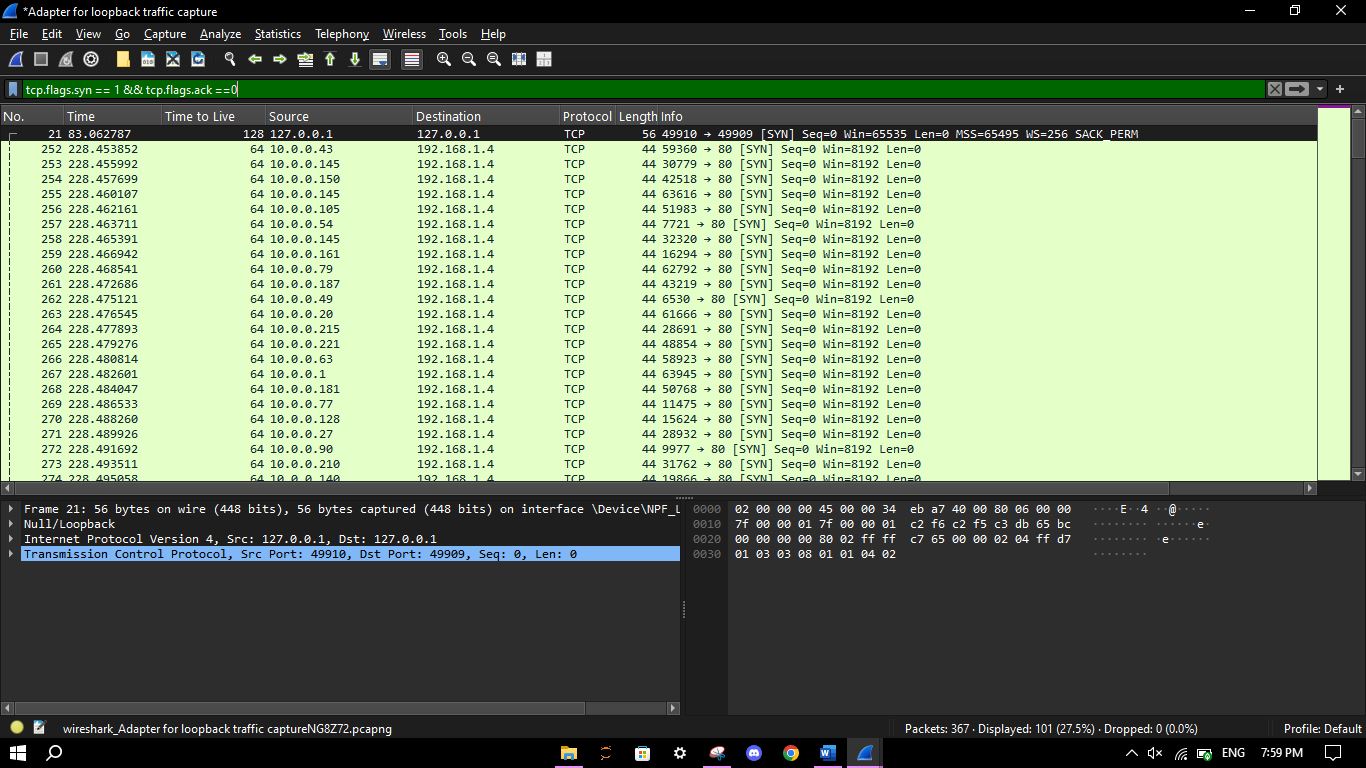






1. All SYN packets

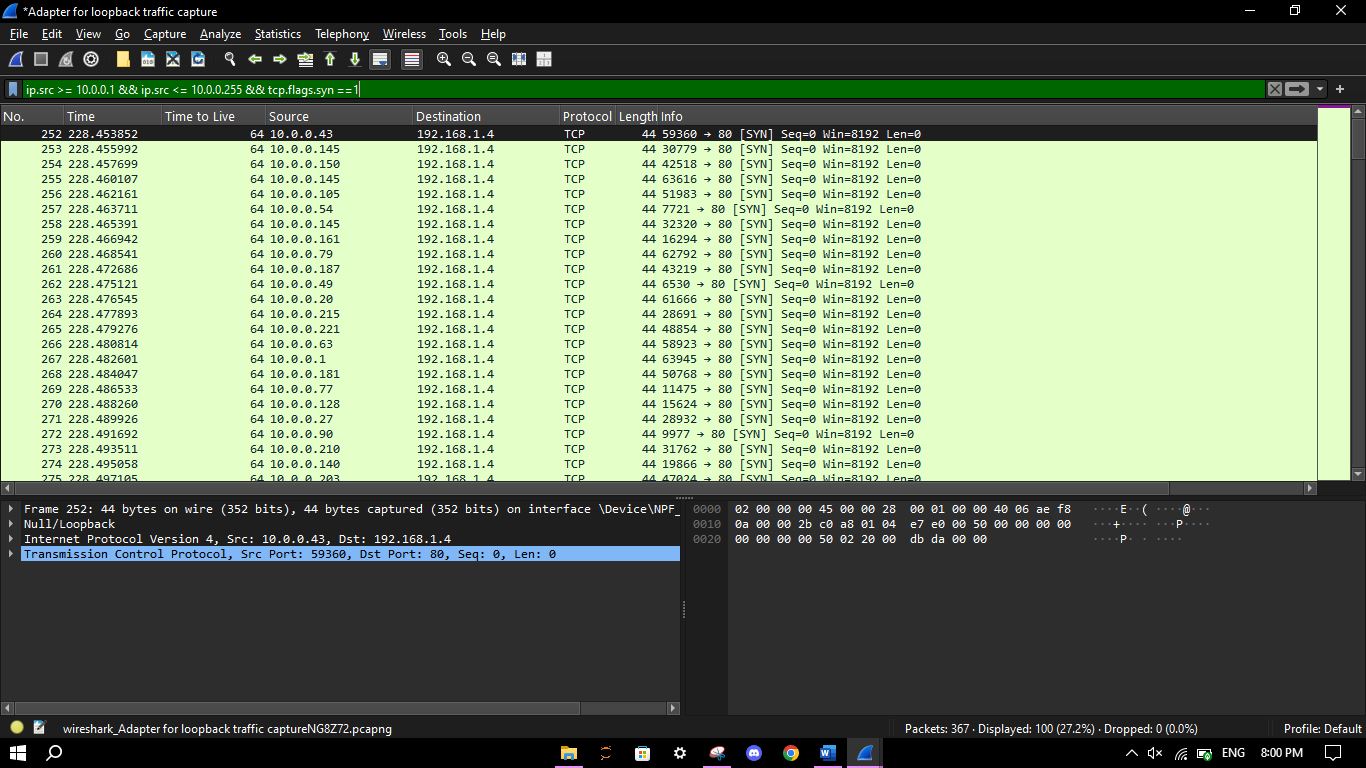
tcp.flags.syn == 1 && tcp.flags.ack == 0





1. All SYN packets from the source “10.0.0”

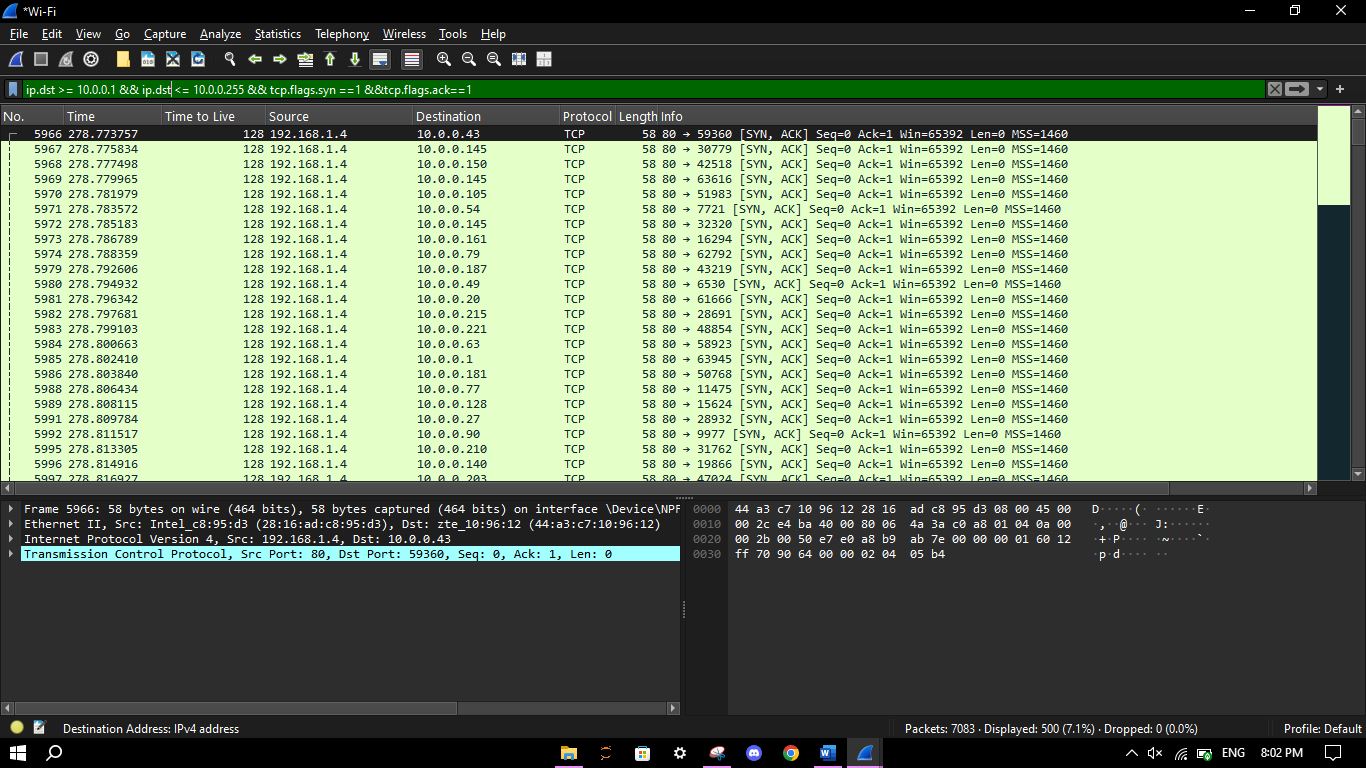
ip.src >= 10.0.0.1 && ip.src <= 10.0.0.255 && tcp.flags.syn == 1 && tcp.flags.ack == 0





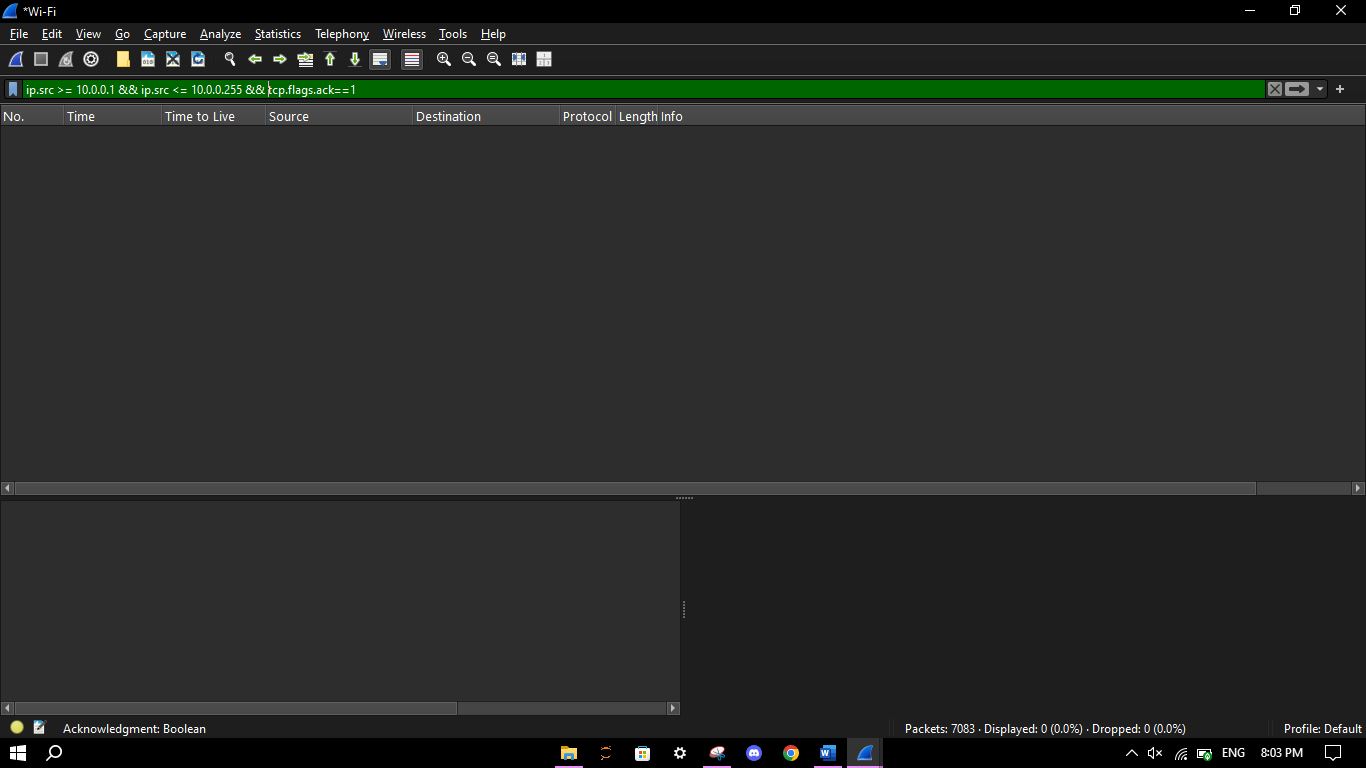
1. All SYN and ACK packets sent to the destination “10.0.0”

ip.dst >= 10.0.0.1 && ip.dst <= 10.0.0.255 && tcp.flags.syn == 1 && tcp.flags.ack == 1



1. All ACK sent by the source “10.0.0”

ip.src >= 10.0.0.1 && ip.src <= 10.0.0.255 && tcp.flags.syn ==0 && tcp.flags.ack==1

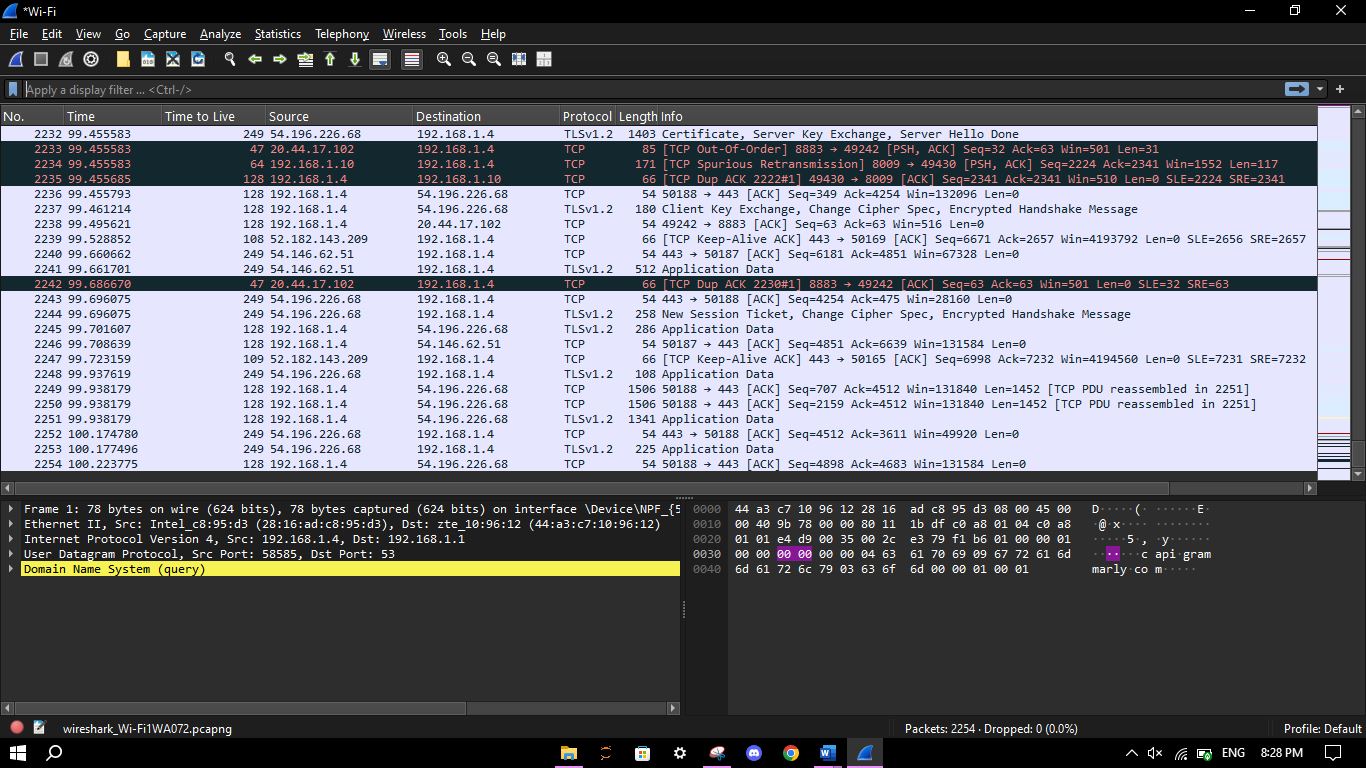


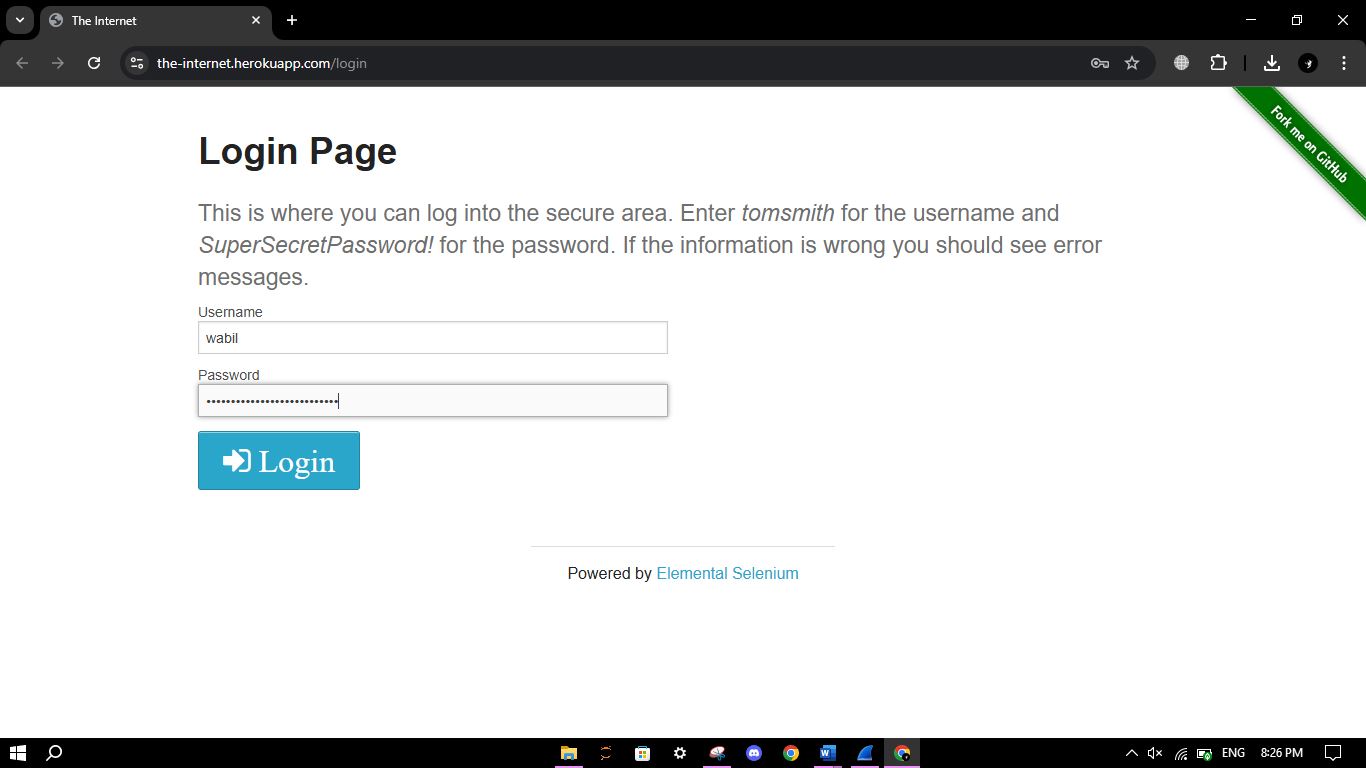
No ACK is returned, confirming SYN flood.

## Use Case 2: HTTPS Header and Payload Decryption

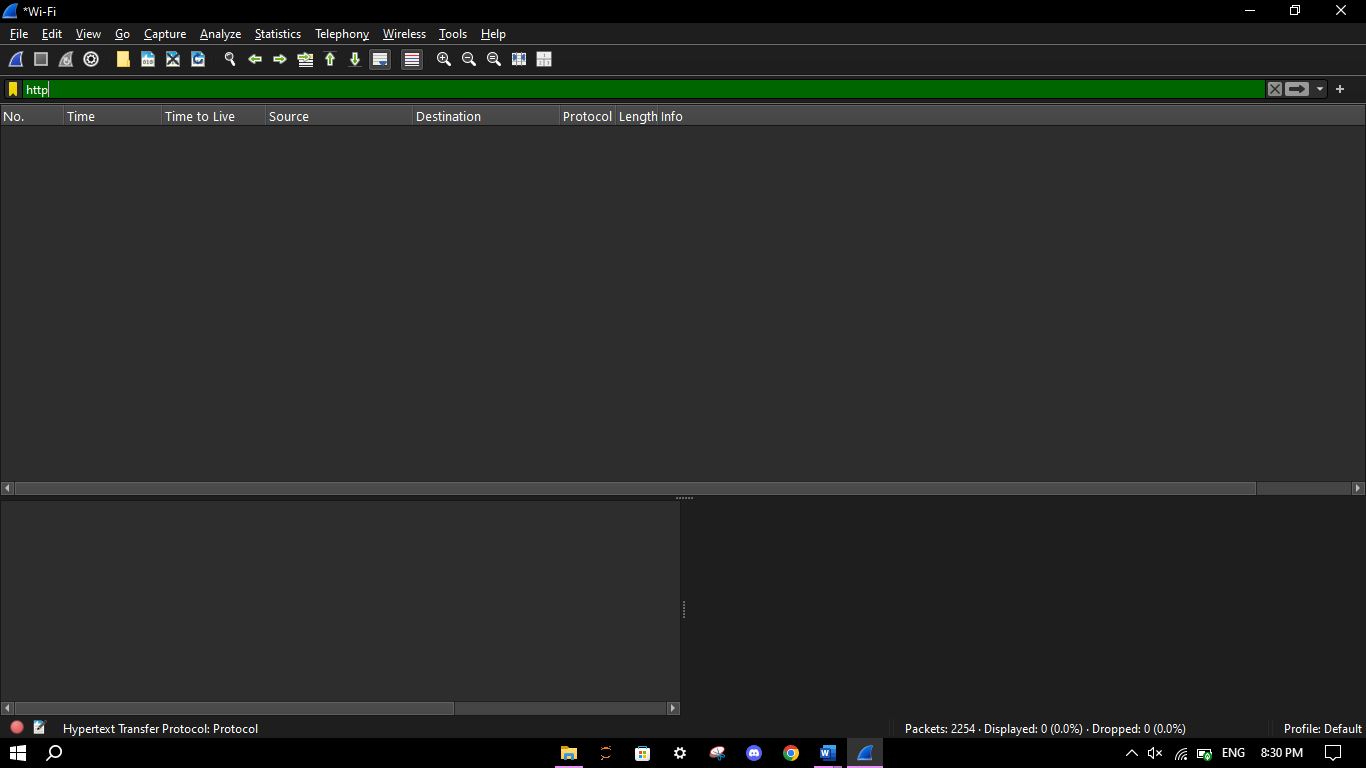
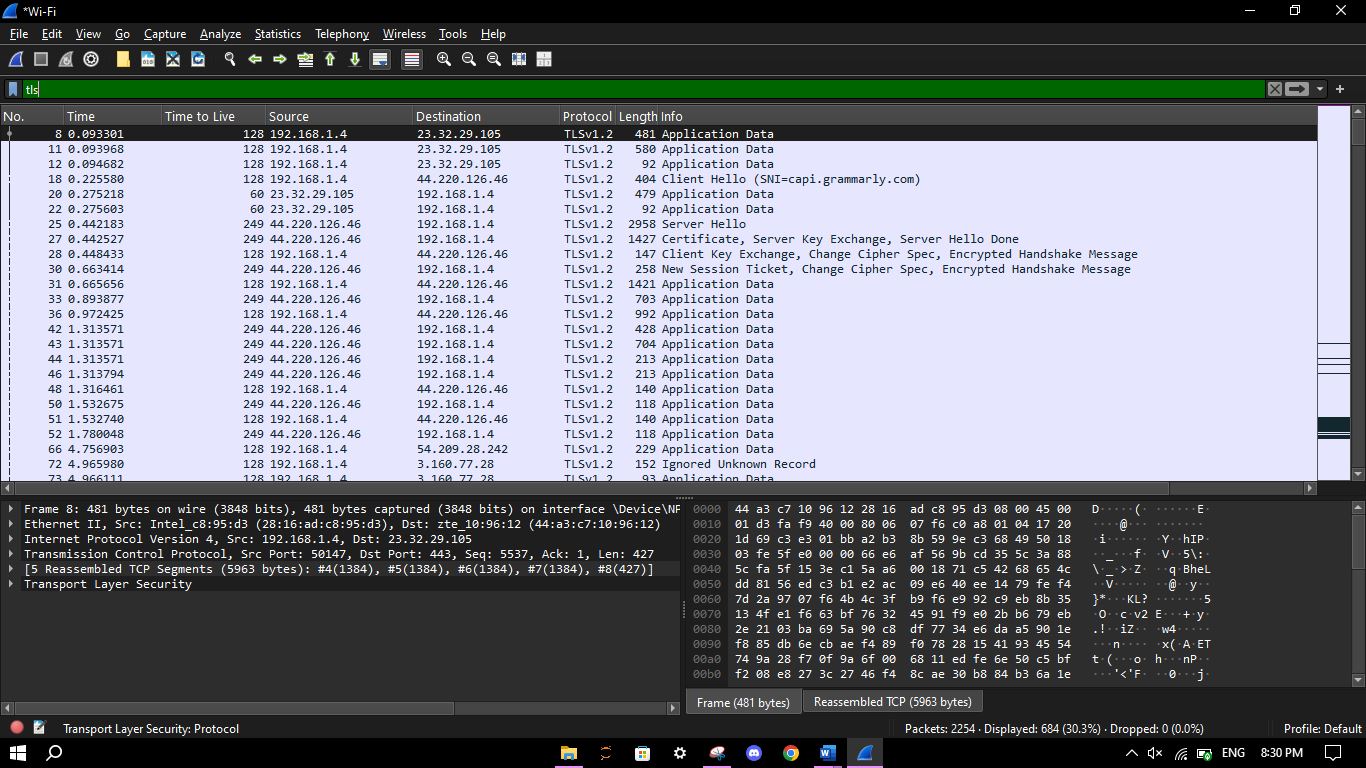
Objective: Use session key logging to decrypt HTTPS traffic in Wireshark.

1. Start Wireshark and go to <https://the-internet.herokuapp.com/login> and login.

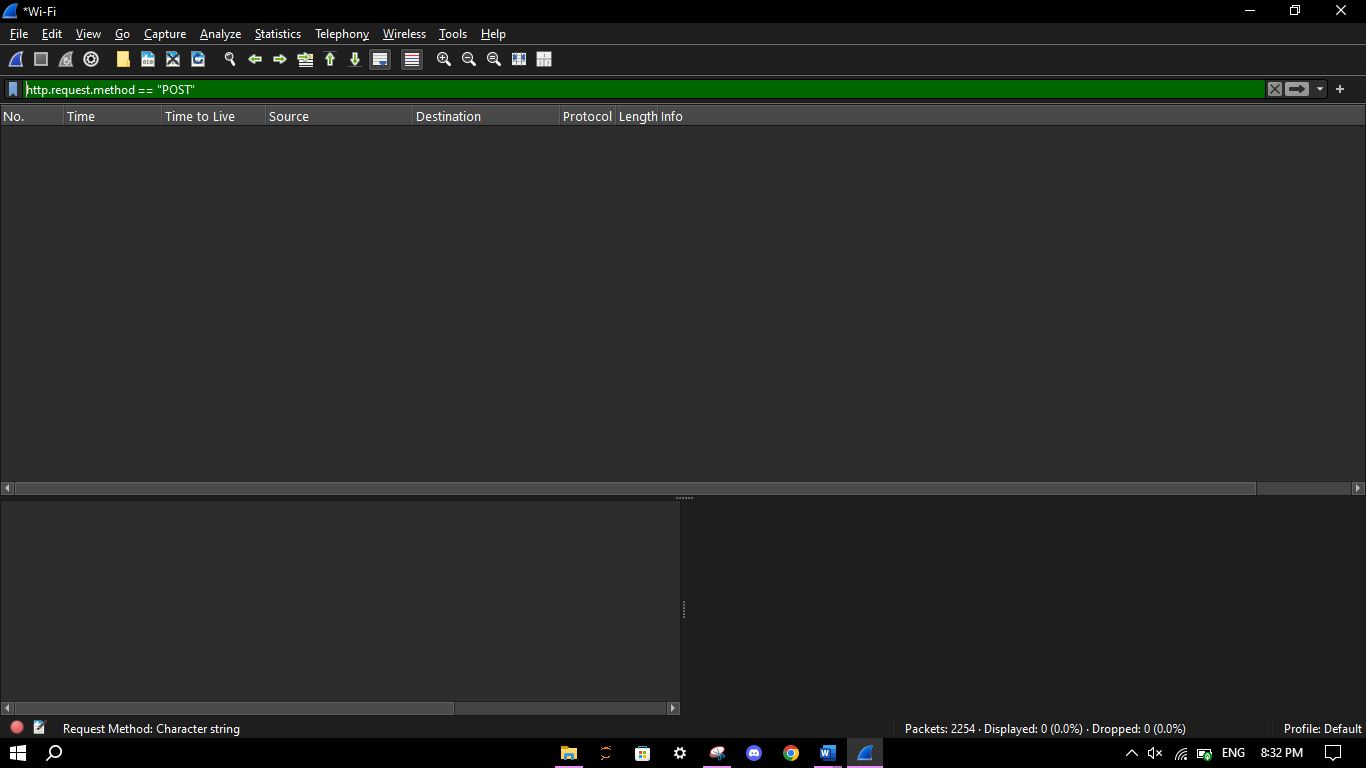




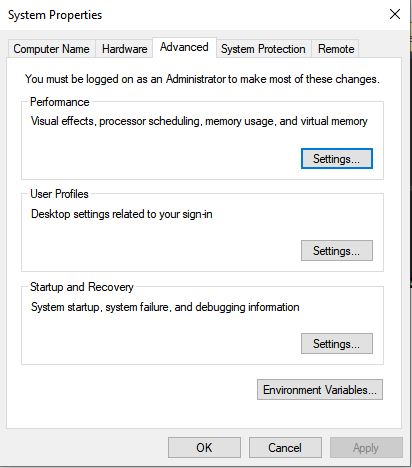
1. Try TLS and HTTP filters on Wireshark to read encrypted data.

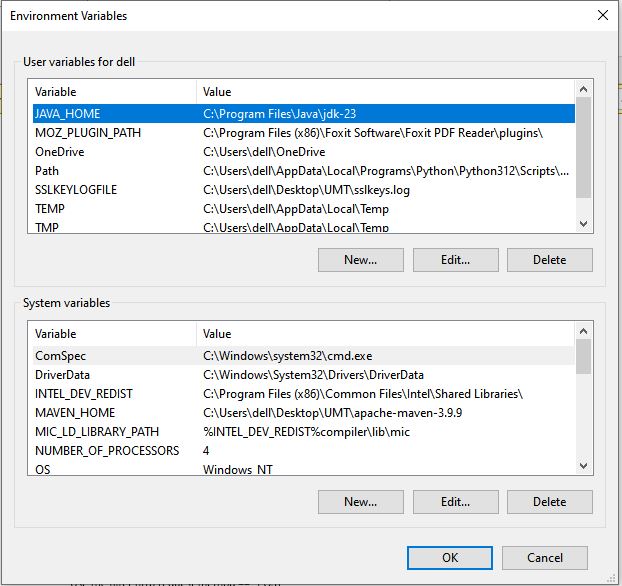


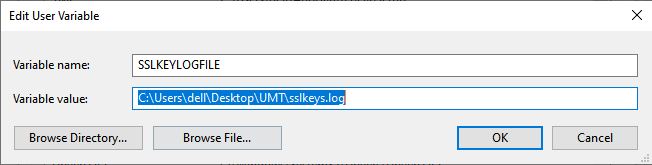
1. Use the filter http.request.method == "POST"



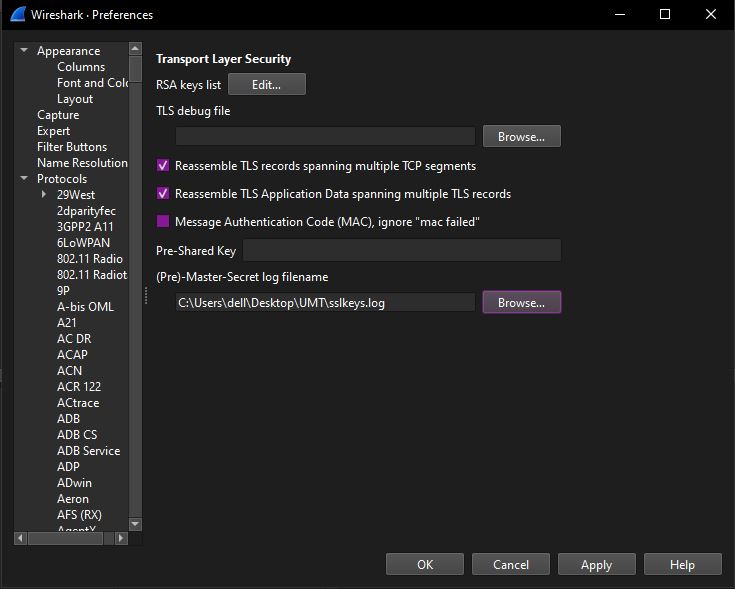
1. Set SSLKEYLOGFILE environment variable in the browser.



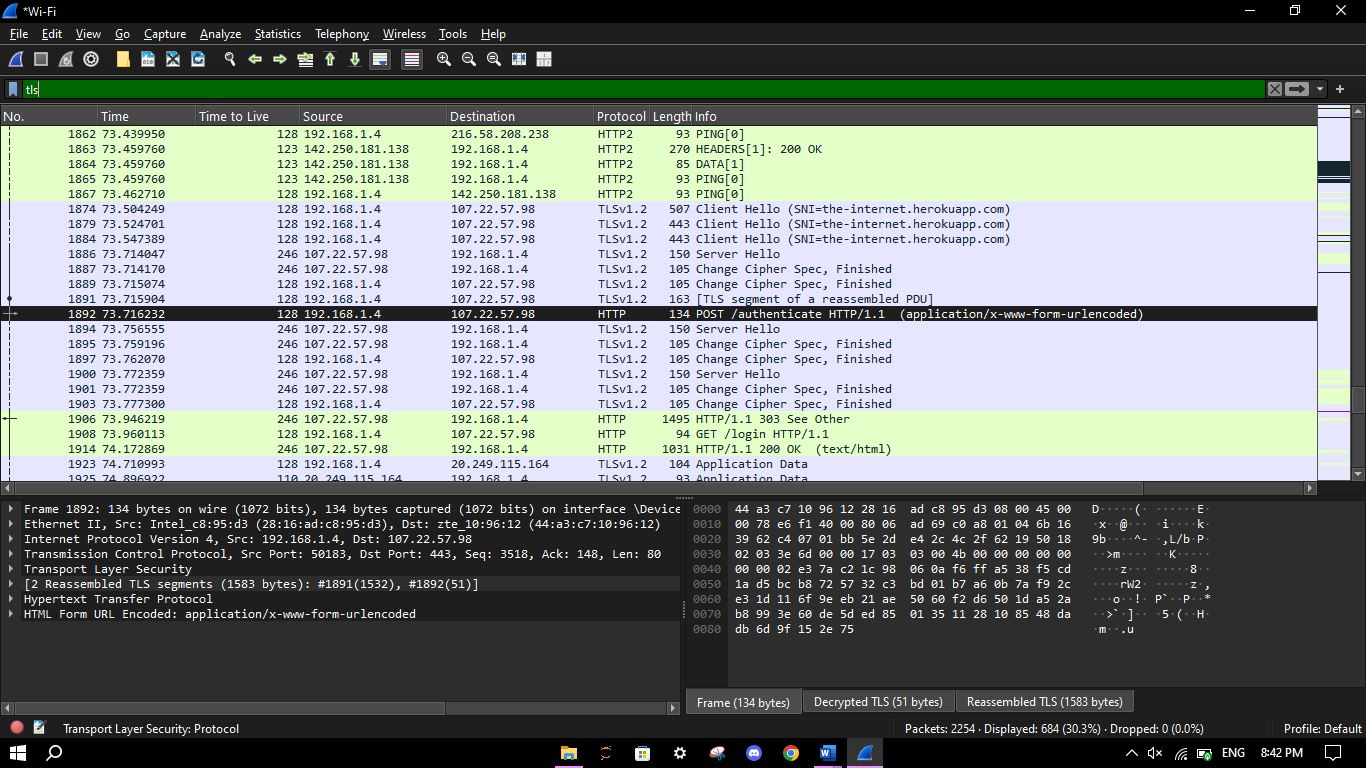




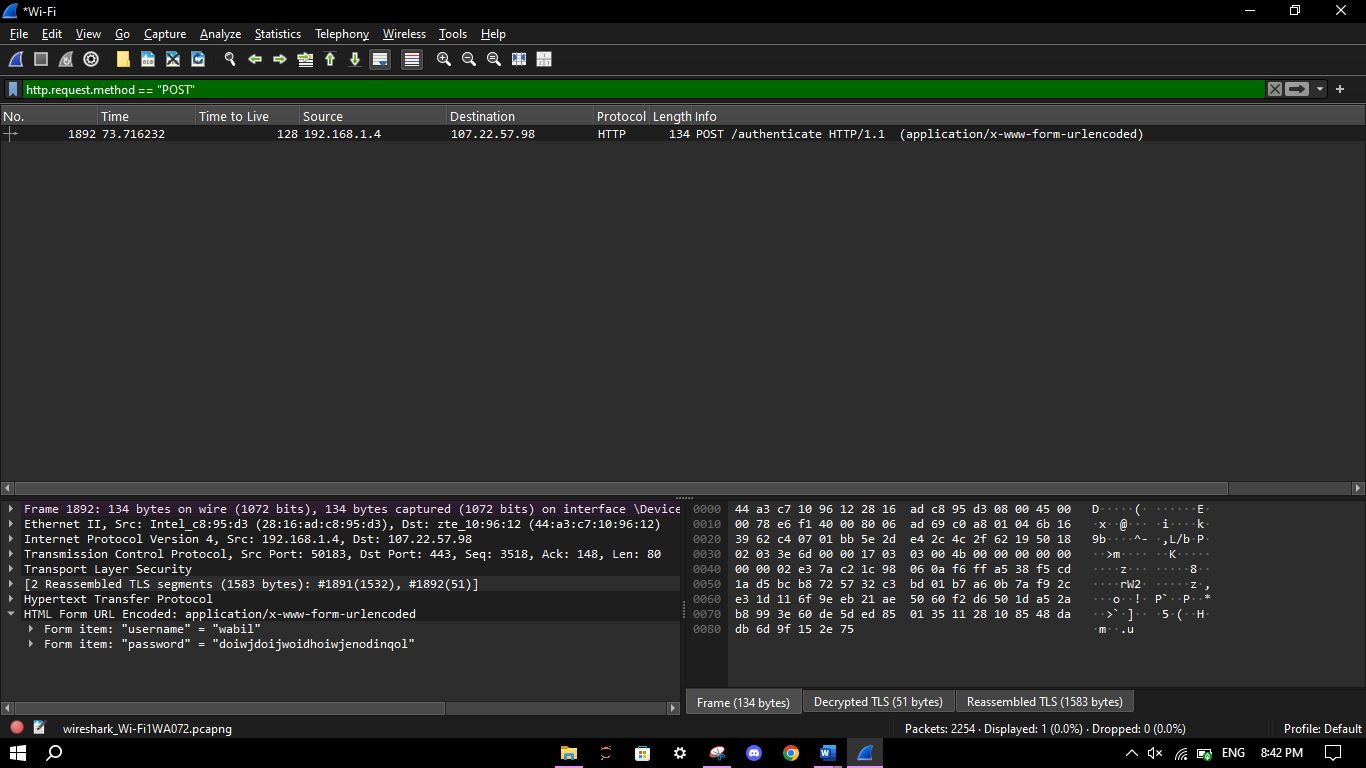
1. Load SSL key file under Preferences → Protocols → TLS. Set (Pre)-Master-Secret log filename to your sslkeys.log file.



1. Try TLS and HTTP filters on Wireshark to read encrypted data.



1. Use the filter http.request.method == "POST"

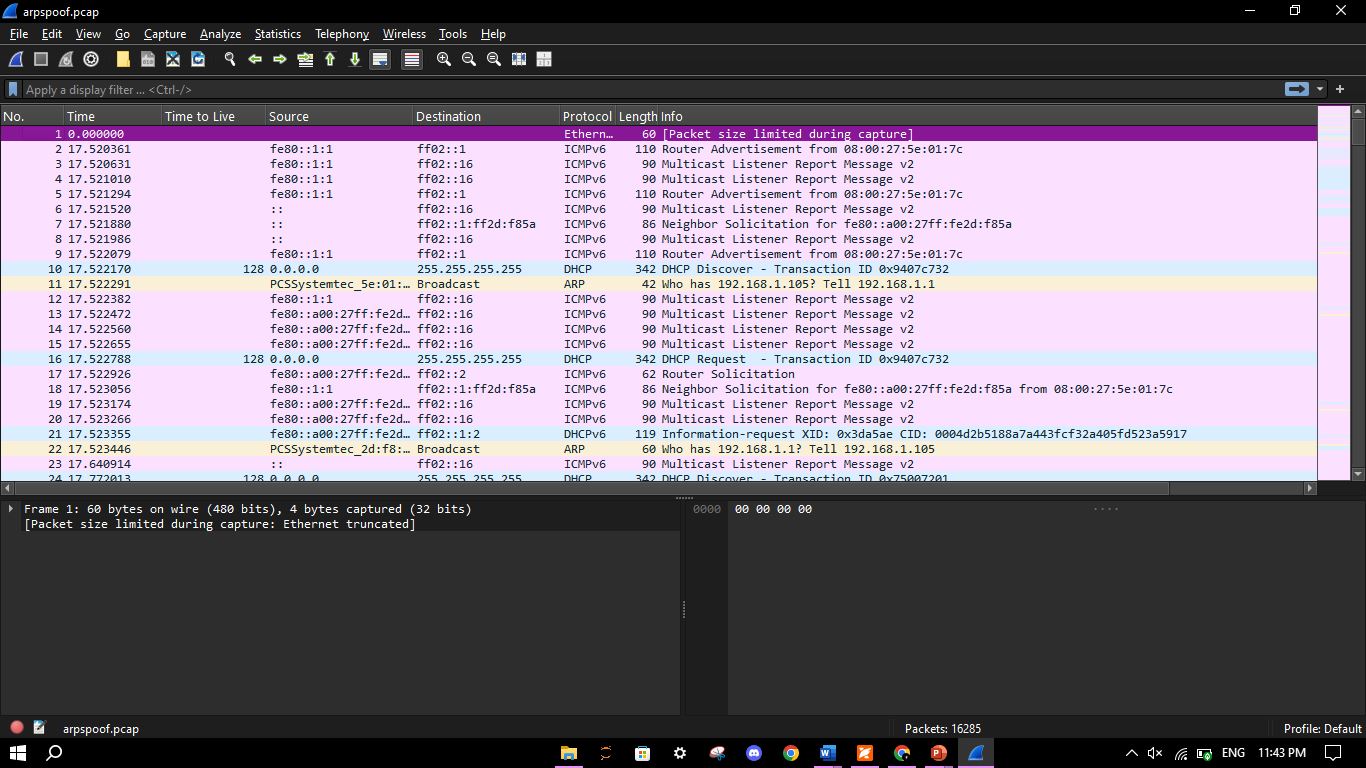


Decrypted and inspected the payload using session keys.

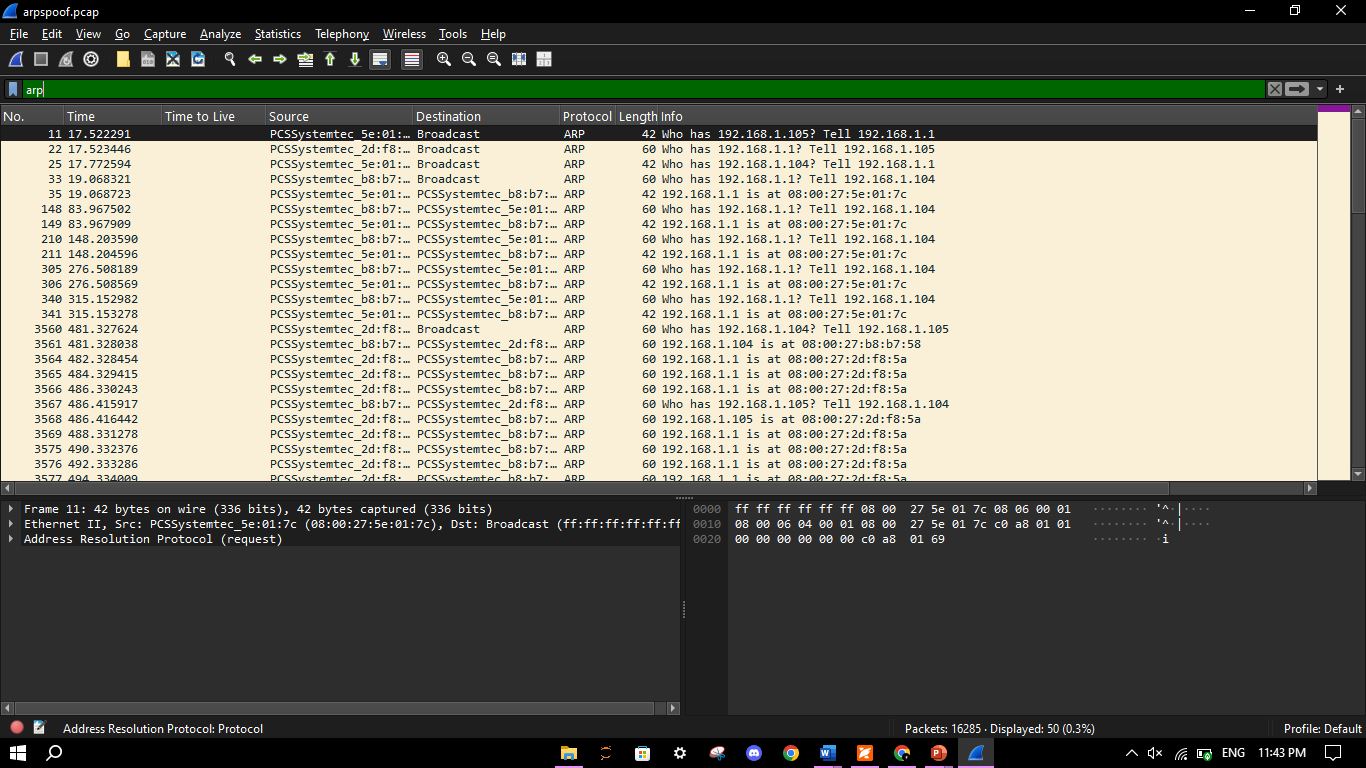
## Use Case 3: ARP Poisoning Detection using Sample PCAP

Objective: Capture and analyze ARP traffic to detect an ARP spoofing attack.

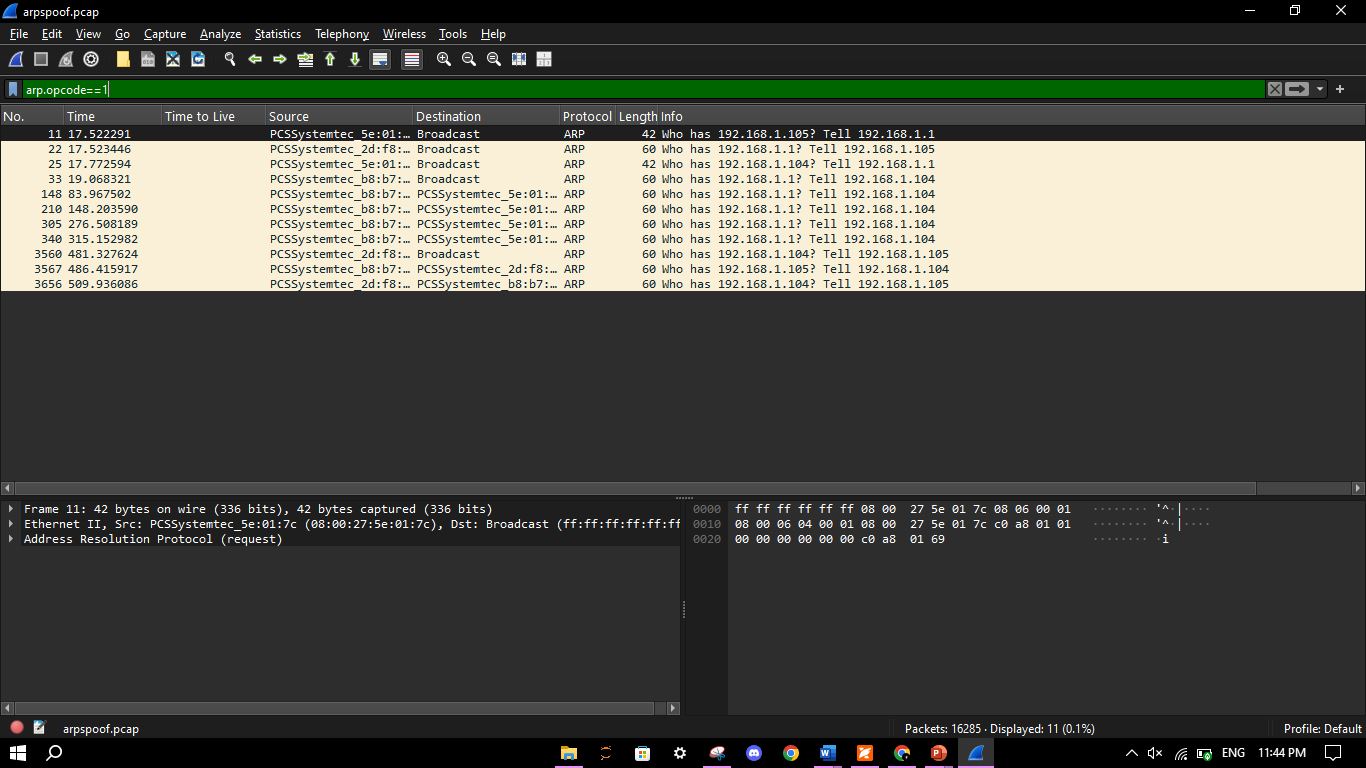
1. Open Wireshark and load arpspoof.pcap file (from GitHub).



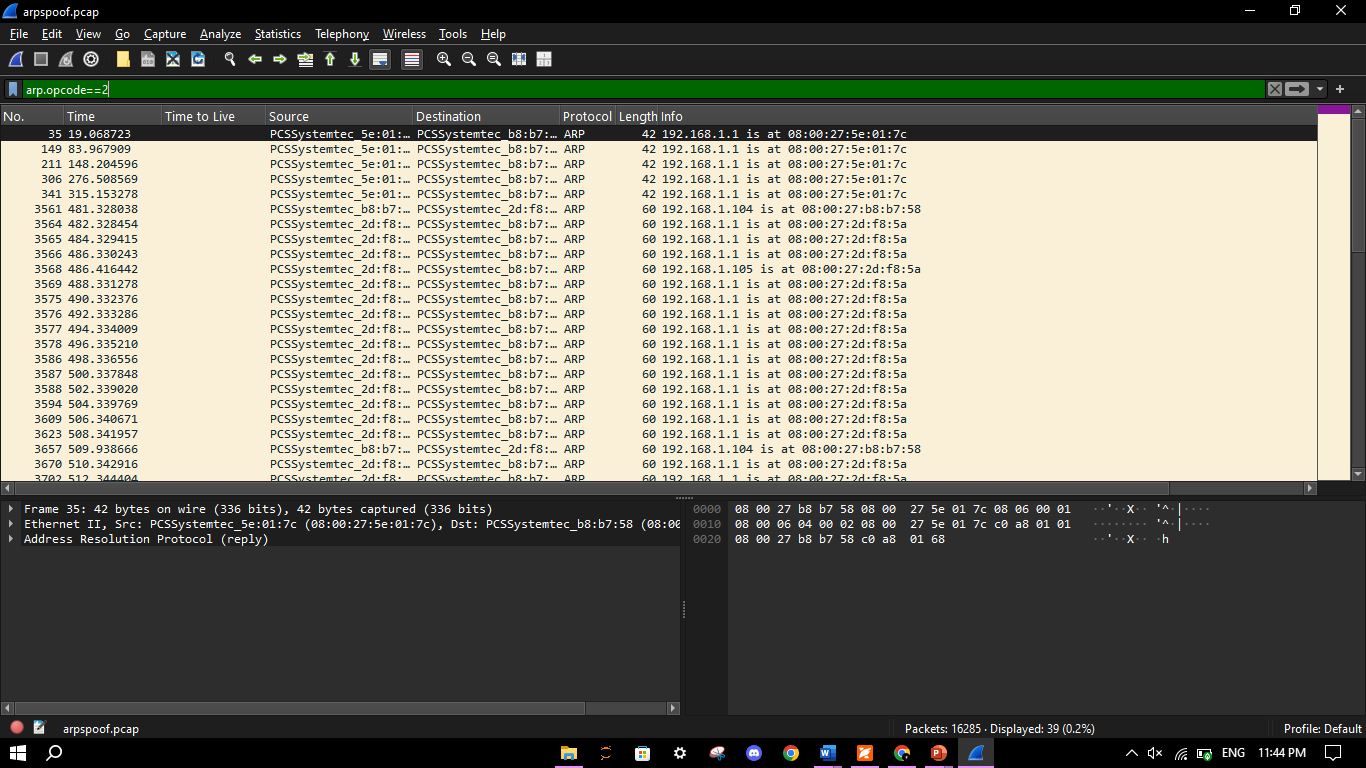
1. Apply ARP filter: arp



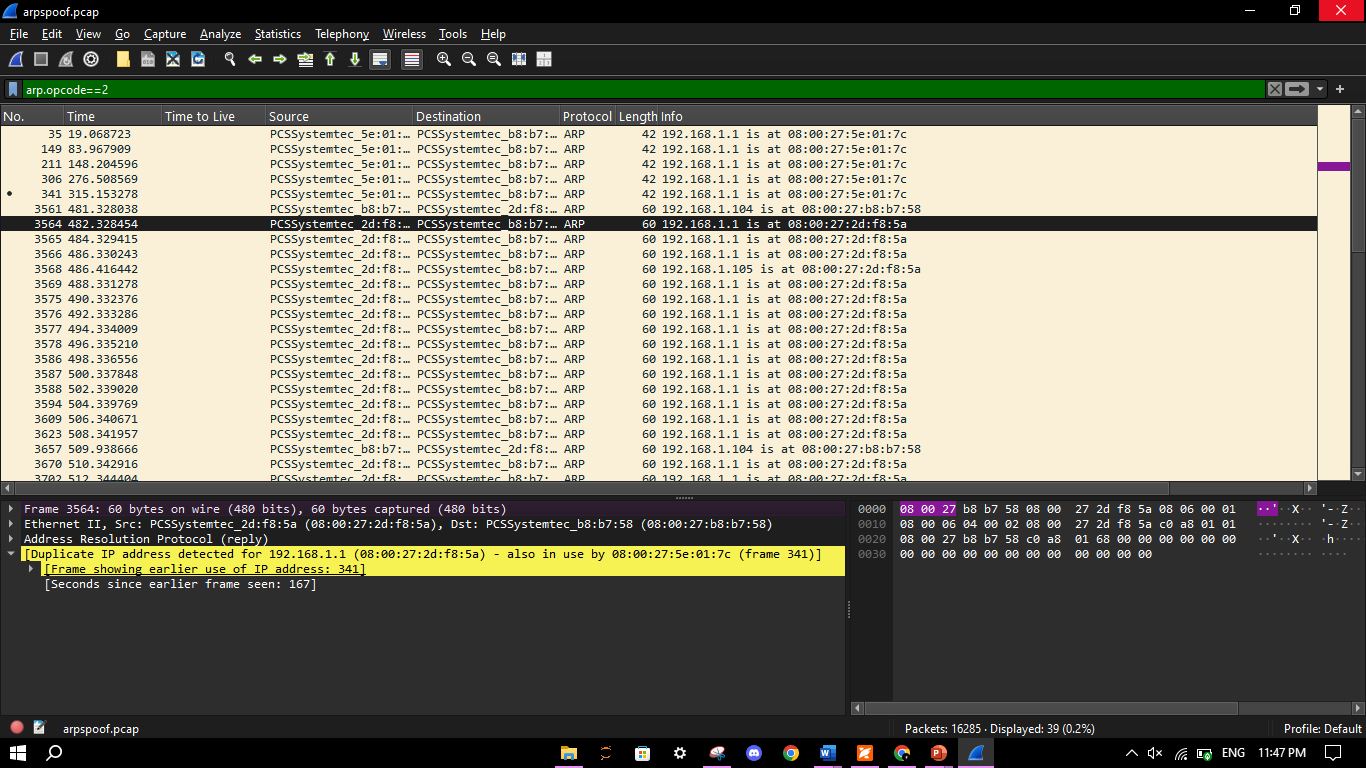
1. Use specific filter: arp.opcode == 1 to isolate ARP broadcasts.



1. Use specific filter: arp.opcode == 2 to isolate ARP replies.



1. Detect change: IP 192.168.1.1 maps to two different MACs in separate replies.



Wireshark shows a warning: "Duplicate IP address detected."

# OSI Layer Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Tool | Use Case | OSI Layer | Role in Security |
| Wireshark | SYN Flood Detection | **Transport Layer (Layer 4)** | Monitors TCP handshake for SYN flood anomalies |
| Wireshark | HTTPS Decryption via Key Log File | **Application Layer (Layer 7)** | Exposes application-level data after SSL/TLS decryption |
| Wireshark | ARP Poisoning Detection | Data Link Layer (L2) | Detects MAC address spoofing in ARP replies |

# MITRE ATT&CK Mapping

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case | Tool Feature | MITRE Tactic | Technique | Technique ID |
| SYN Flood Detection | TCP SYN packet capture | **Impact** | **Direct Network Flood** | **T1498.001** |
| HTTPS Decryption (simulated) | SSL key file decryption | **Credential Access** | **Unsecured Credentials – Session Keys** | **T1552.004** |
| ARP Spoofing Detection | ARP reply conflict analysis | Credential Access / Collection | ARP Cache Poisoning | T1200 |

# 6. Mitigation Strategies

|  |  |
| --- | --- |
| Risk | Mitigation |
| **Unauthorized Packet Sniffing** | Ensure Wireshark is installed and run only by **trusted admin users**. Require **admin/root privileges** to capture traffic. |
| **HTTPS Decryption Abuse** | Implement **Perfect Forward Secrecy** (e.g., ECDHE), so session keys cannot be reused. Never store SSL key log files in production. |
| **Insider Threat or Lab Abuse** | Monitor internal systems for unauthorized Wireshark activity. Use **SIEM tools** or host-based logging to detect captures. |
| **Data Exfiltration via Captured Traffic** | Restrict **upload/email access** from machines where packet captures are allowed. Encrypt or sanitize PCAP files before sharing. |

# 7. References

* Wireshark Documentation: <https://www.wireshark.org/>
* Scapy Library: <https://scapy.readthedocs.io/>
* MITRE ATT&CK Framework: <https://attack.mitre.org/>
* SSL Decryption: <https://wiki.wireshark.org/TLS>
* ARP Spoofing PCAP Sample: <https://github.com/researcher111/ARP-pcap-files/blob/master/arpspoof.pcap>