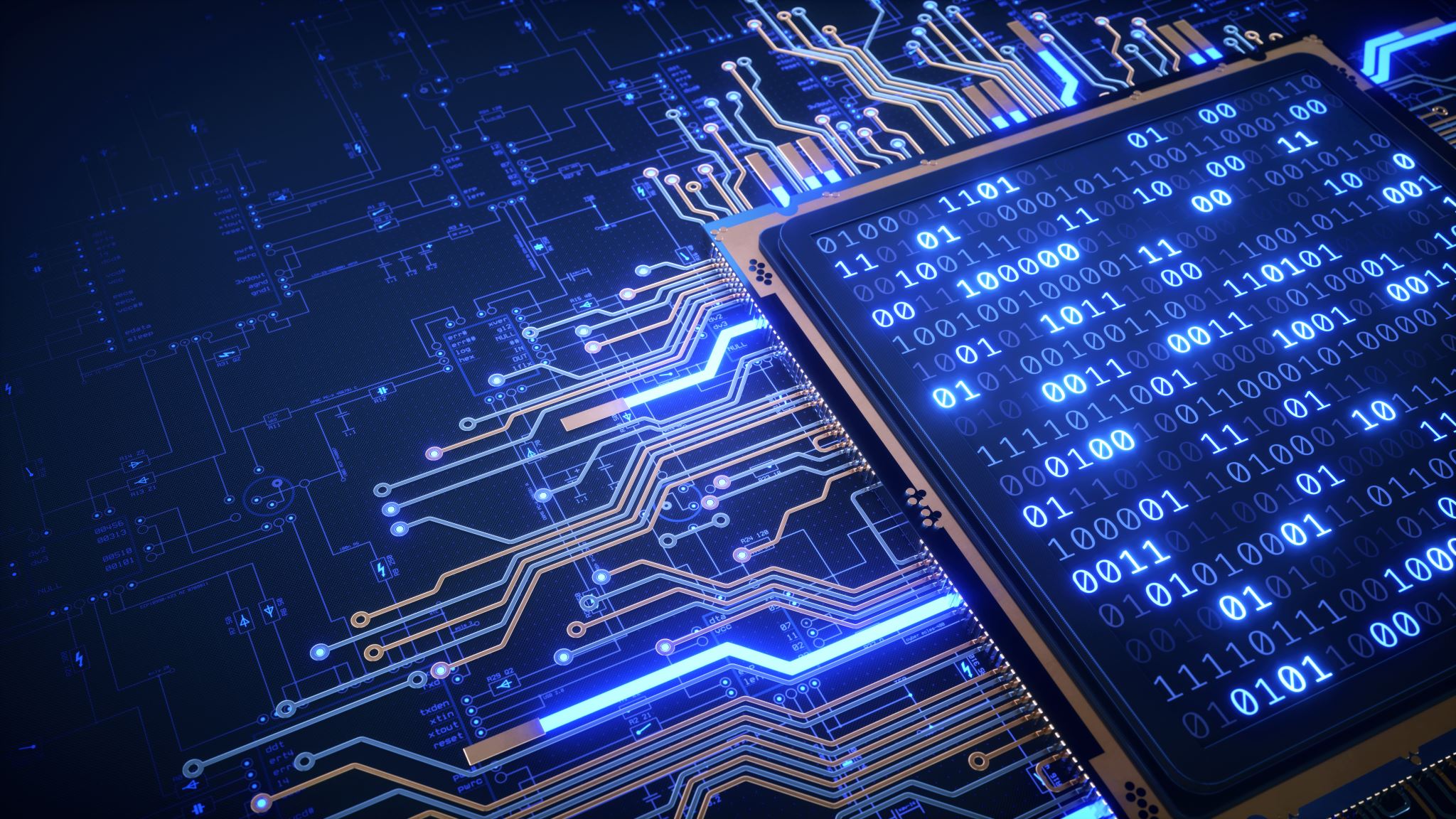
***Network Traffic Analysis Using Wireshark and Zeek***Project Report  
Submitted By: Garima Jaiswal

Under the Guidance of: MR. Nikhil Pandey  
Course: Cyber Security

Institution: United College Of Engineering and Research, Prayagraj  
Date: 29/07/2025

*Abstract*



In today's digital world, network security and performance have become crucial concerns for organizations and individuals. Analyzing network traffic helps in detecting intrusions, identifying anomalies, optimizing bandwidth usage, and improving overall network performance. This project focuses on using two powerful open-source tools — Wireshark and Zeek (formerly Bro) — for network traffic analysis.  
  
Wireshark offers deep packet inspection, protocol analysis, and filtering capabilities, while Zeek enables scripting-based analysis for real-time detection of suspicious behavior. This report presents a comparative and practical study of both tools by capturing and analyzing traffic from various simulated and real network environments. The report highlights the methodologies, results, and best practices for leveraging both tools effectively for cybersecurity and traffic management.

# *Table of Contents*

1. Title Page

2. Abstract

3. Table of Contents

4. List of Figures and Tables

5. Introduction

6. Literature Review

7. Methodology/Approach

8. Results and Discussion

9. Detection of Network Threats Using Wireshark

10. Detection of Network Threats Using Zeek

11. Conclusion

12. Recommendations

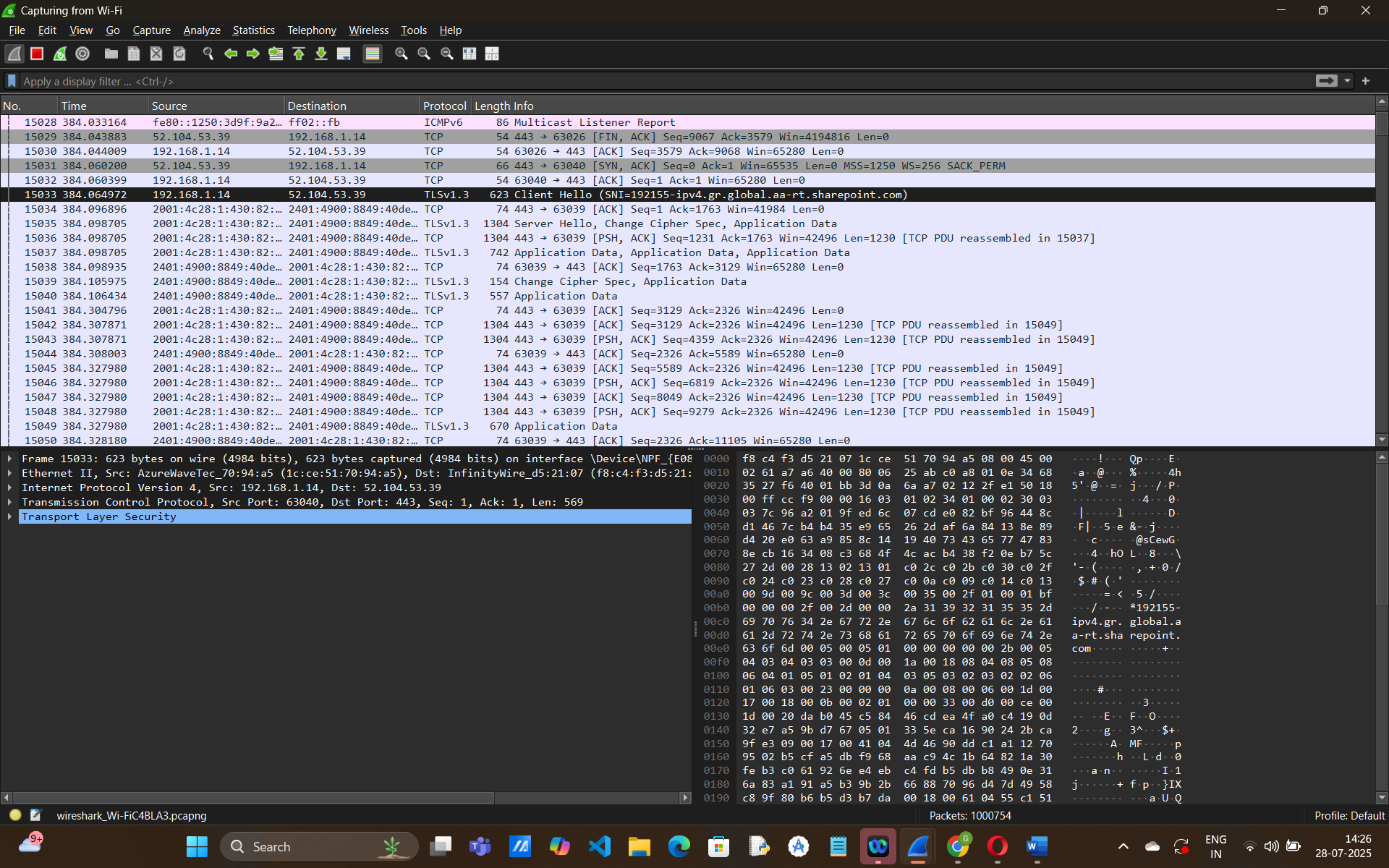
13. References

14. Appendices

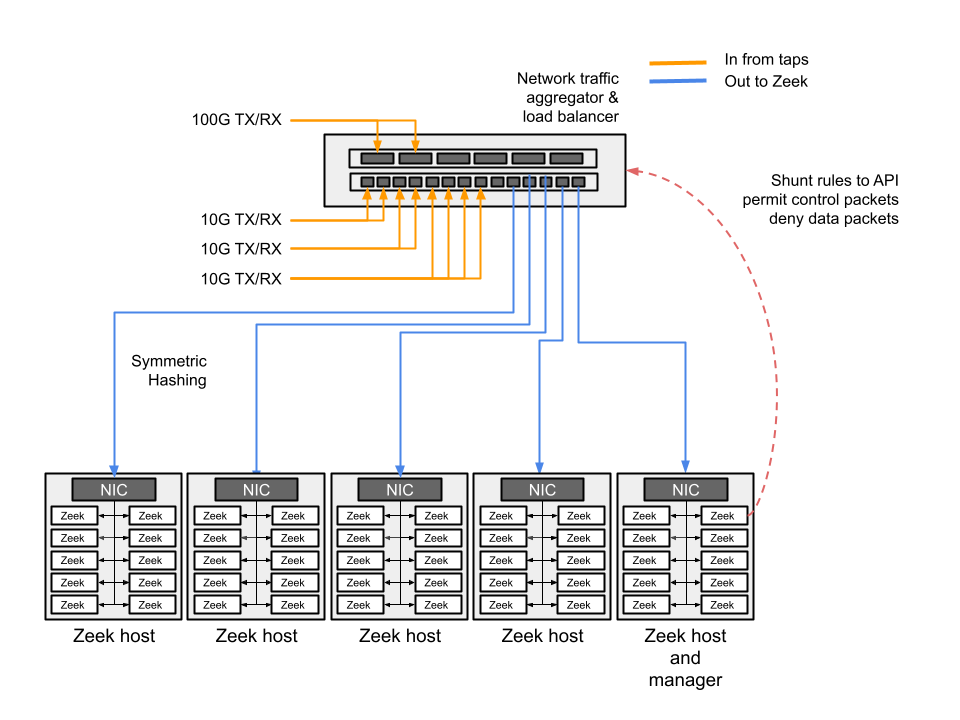
# *List of Figures and Tables*

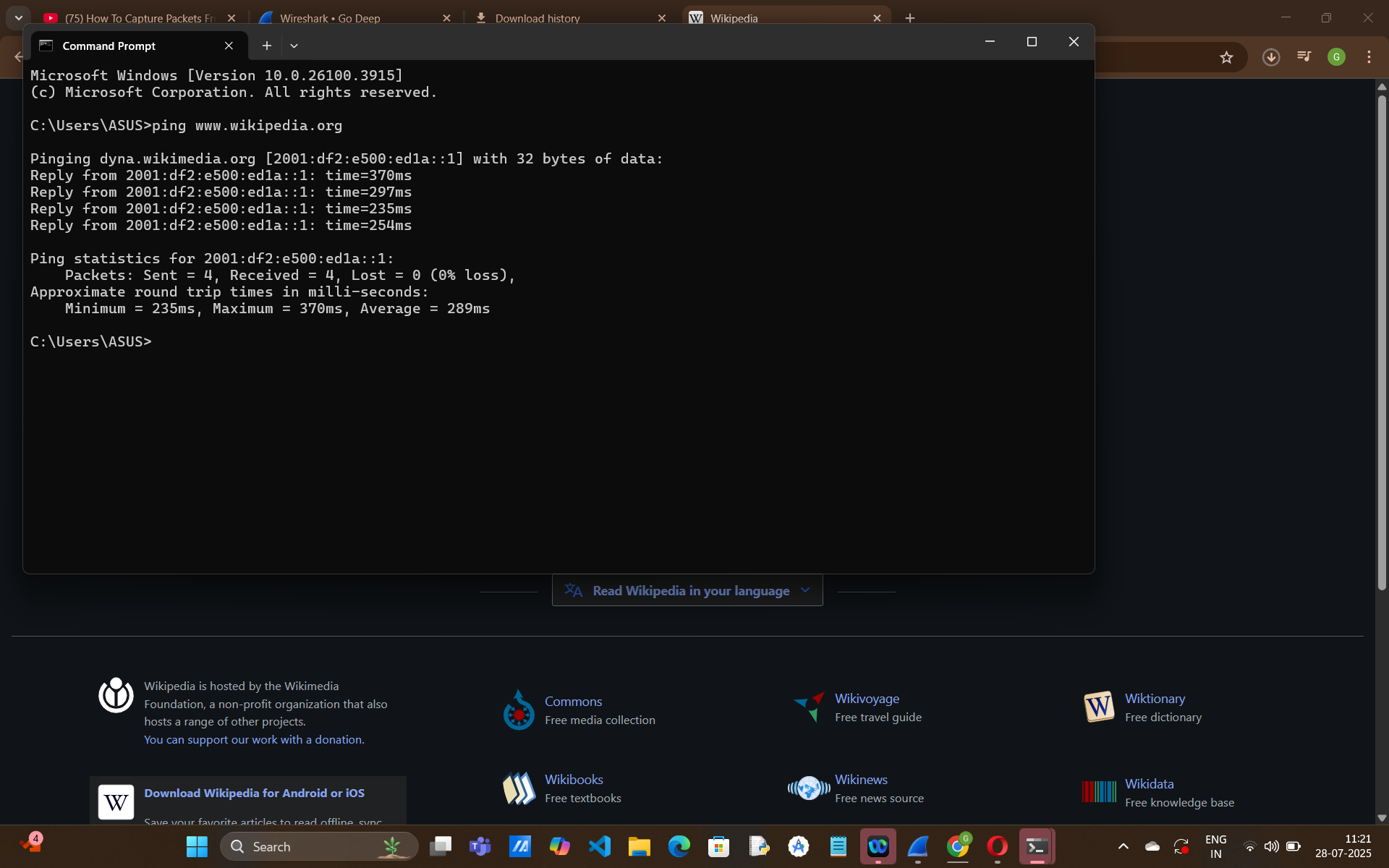
***Figure 1: Wireshark User Interface***

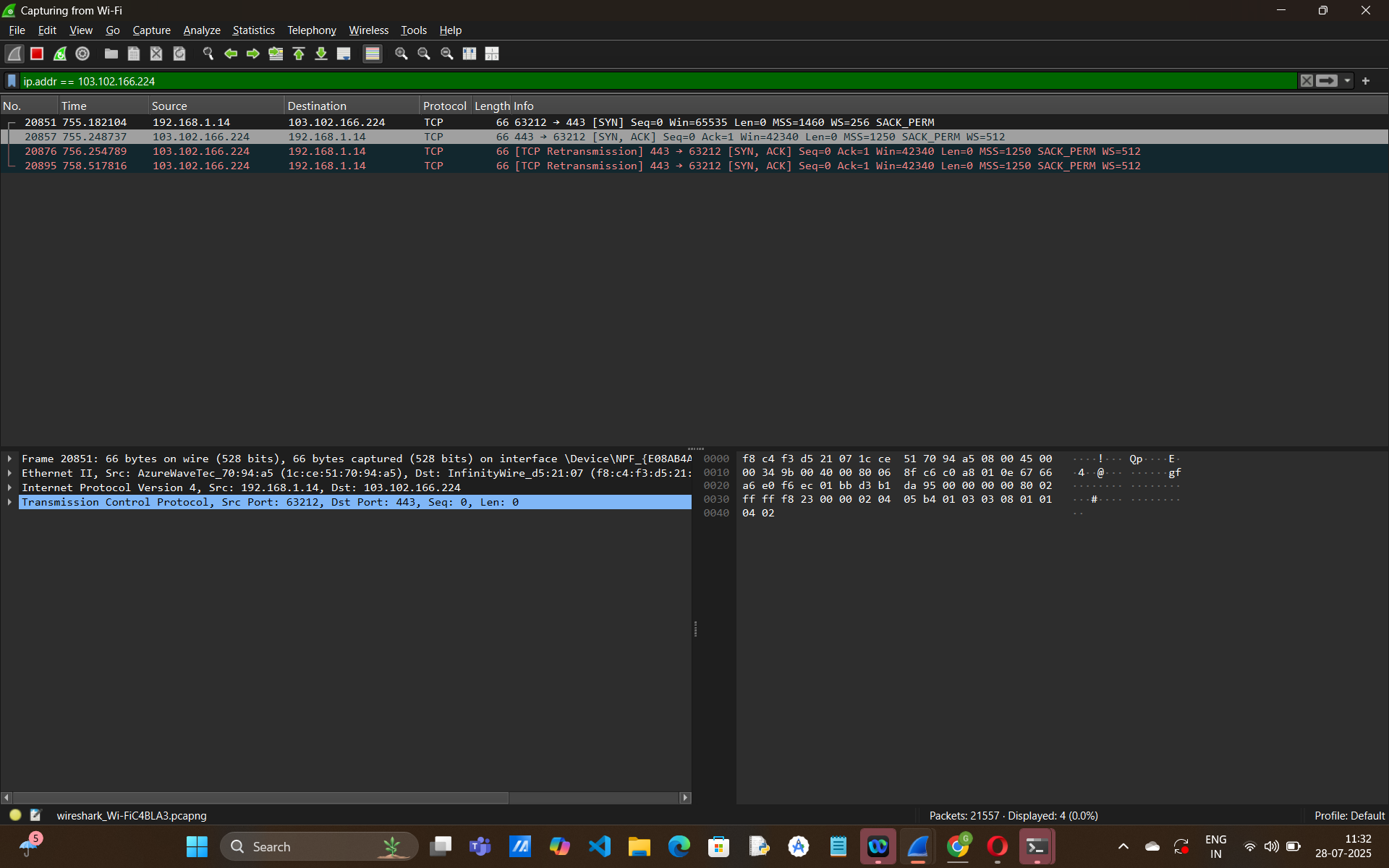


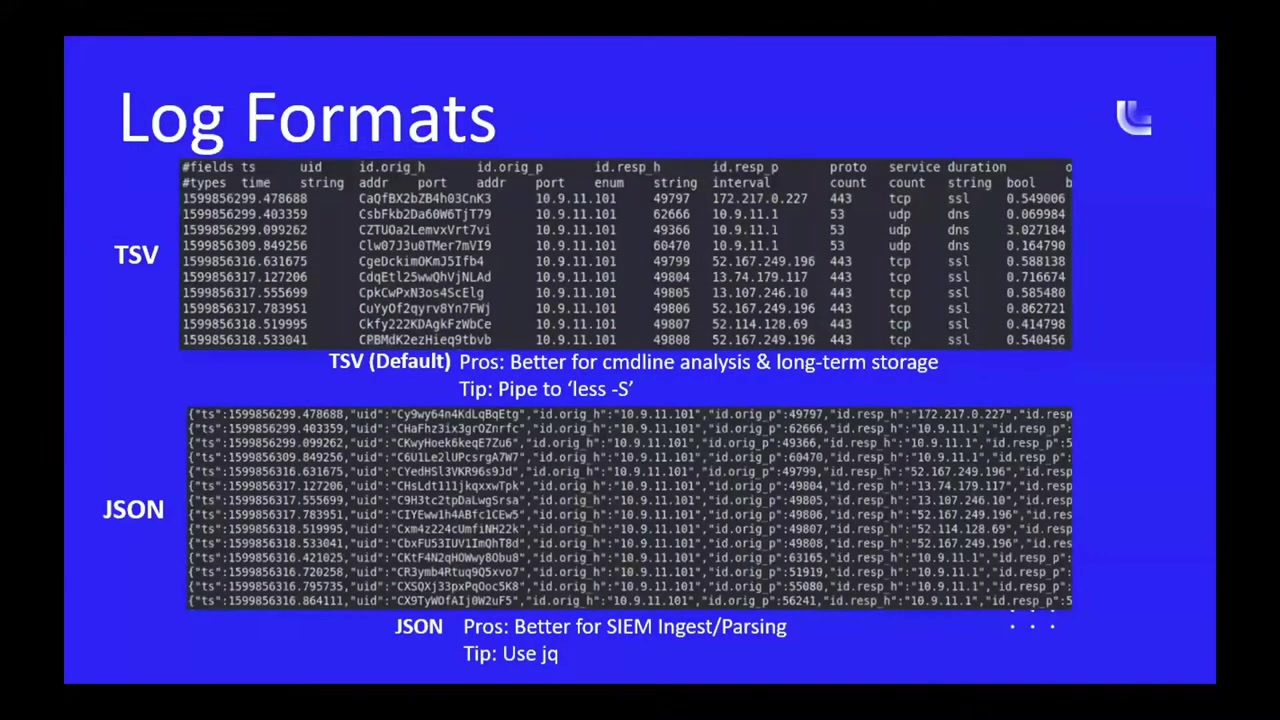


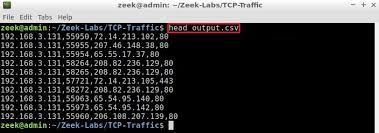
***Figure 2: Zeek Architecture Diagram***

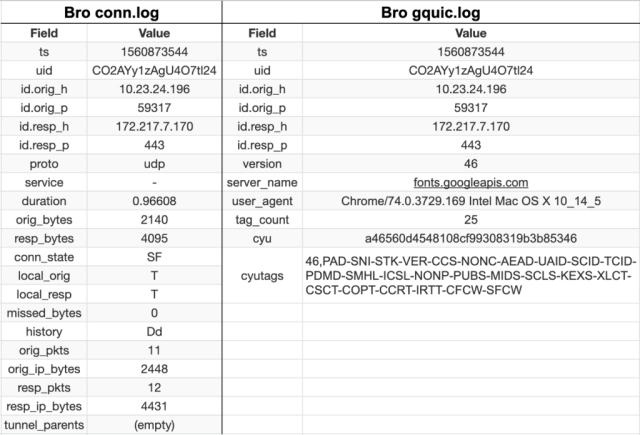


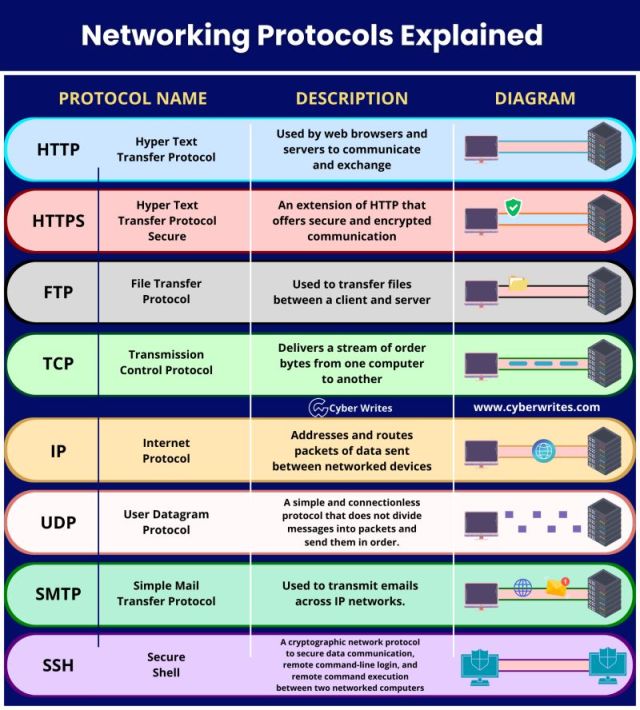
***Figure 3: Packet Capture Example in Wireshark***



***Figure 4: Zeek Log Files Overview***



***Table 1: Comparison Between Wireshark and Zeek***

***Table 2: Common Protocols Analyzed***

***Table 3: Captured Traffic Summary***

# *Introduction*



With the increasing dependence on digital communication, monitoring and analyzing network traffic has become critical. Whether it is for detecting unauthorized access, debugging performance issues, or enforcing policies, network traffic analysis (NTA) plays a pivotal role.  
  
This project uses Wireshark, a GUI-based packet sniffer, and Zeek, a script-based network monitoring framework, to capture, monitor, and analyze traffic patterns. The goal is to understand the behavior of various protocols, detect anomalies, and improve network visibility.

# Literature Review



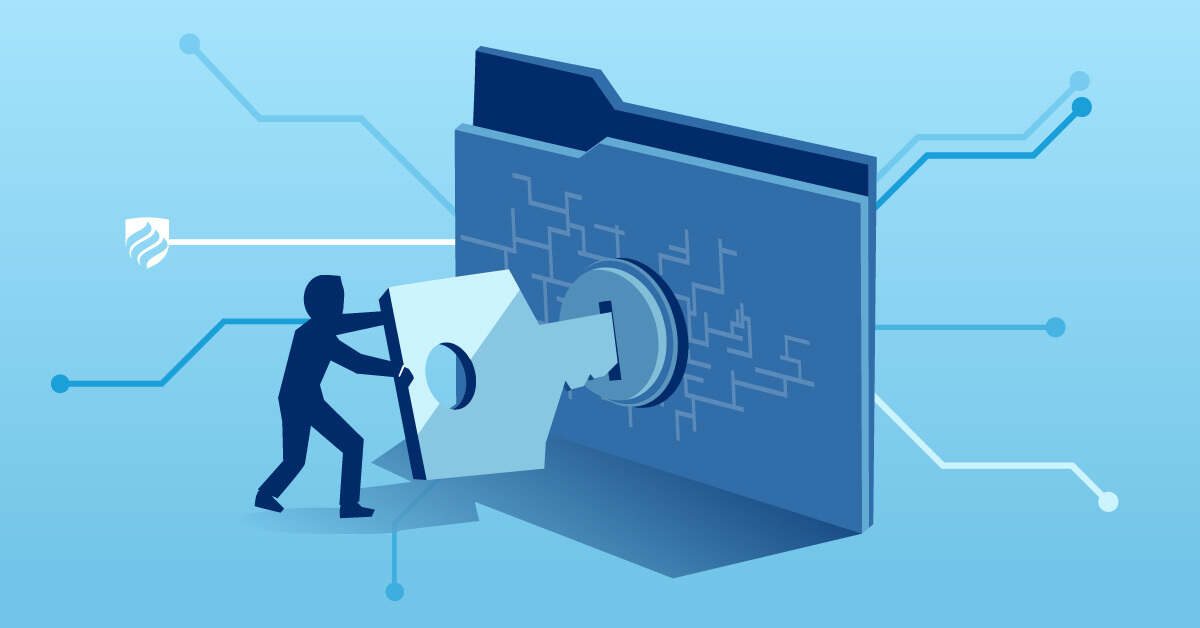
Several studies have explored network traffic analysis for cybersecurity and optimization:  
  
- Abdelsalam et al. (2020) emphasized the importance of anomaly detection using flow-based analysis.  
- Wireshark has been used in academia and industry for protocol education and forensic analysis.  
- Zeek, with its extensibility through scripting, has been integrated with SIEM systems and intrusion detection pipelines.  
- Recent research highlights the effectiveness of combining multiple tools to enhance detection accuracy and reduce false positives.

# Methodology/Approach



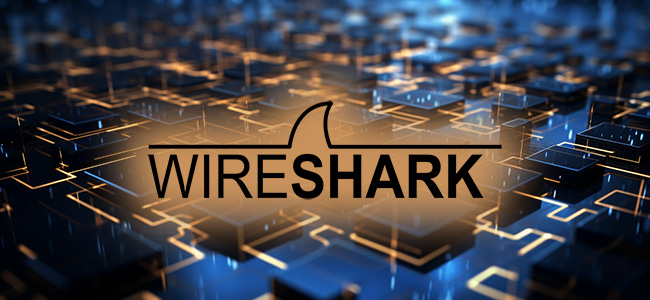
3.1 Tools Used  
- Wireshark – For packet-level capture and deep inspection.  
- Zeek – For high-level event-driven analysis and log generation.  
  
3.2 Environment Setup  
- Ubuntu 22.04 (Virtual Machine)  
- Sample network traffic from pcap files and live capture from LAN  
- Custom traffic generated using tools like ping, ftp, http traffic  
  
3.3 Process Workflow  
1. Capture live traffic or load `.pcap` files.  
2. Analyze using Wireshark (filters, protocol breakdown).  
3. Run `.pcap` through Zeek to generate logs (conn.log, http.log, dns.log, etc.).  
4. Correlate findings from both tools.  
5. Identify suspicious or anomalous patterns.

# 4. Results and Discussion



4.1 Wireshark Analysis  
- Filtered HTTP traffic: Identified GET/POST requests and headers.  
- Detected ARP spoofing by analyzing broadcast MAC requests.  
- Noticed large number of retransmissions indicating network issues.  
  
4.2 Zeek Analysis  
- Generated comprehensive logs including conn.log, http.log, ssl.log.  
- Detected DNS tunneling and unusual HTTP user-agent strings.  
- Identified login attempts via FTP on non-standard ports.  
  
4.3 Comparative Analysis  
  
| Feature | Wireshark | Zeek |  
|--------|-----------|------|  
| Interface | GUI | CLI/Scripting |  
| Level | Packet | Event-based |  
| Detection | Manual (via filters) | Automated (via scripts) |  
| Logs | Temporary unless saved | Persisted, structured logs |  
| Best Use | Forensic analysis | Real-time detection & alerting |

***5.Detection of Network Threats Using Wireshark***



Focusing on Suspicious Patterns and Protocol Anomalies

**1. Introduction**

With the surge in cyber threats and unauthorized network access, network traffic analysis plays a critical role in cybersecurity. Wireshark, a powerful open-source packet analyzer, enables administrators and analysts to detect threats at the packet level. This report focuses on identifying suspicious patterns, protocol anomalies, and unusual traffic behaviors that signal security threats, and outlines mitigation strategies.

**2. Common Detected Threats in Wireshark**

**2.1 ARP Spoofing / ARP Poisoning**

Pattern: Multiple ARP replies from different MAC addresses for the same IP.

Detection:  
- Apply filter: arp.duplicate-address-detected == 1  
- Look for unsolicited ARP replies or abnormal broadcast storms.

**2.2 DNS Tunneling**

Pattern: Abnormally long DNS queries or frequent unusual domain requests.

Detection:  
- Filter: dns.qry.name contains "suspiciousdomain"  
- Look for repeated TXT records or uncommon domains.

**2.3 SYN Flood (DoS Attack)**

Pattern: Excessive SYN packets with no corresponding ACK.

Detection:  
- Filter: tcp.flags.syn == 1 and tcp.flags.ack == 0  
- Use “Statistics > Conversations” to track large volumes from single source IPs.

**2.4 Port Scanning**

Pattern: Numerous connection attempts to different ports from the same source.

Detection:  
- Filter: tcp.flags.syn == 1  
- Use “Statistics > Endpoints” and inspect the destination port variety.

**2.5 Unencrypted Credentials**

Pattern: HTTP POST requests containing usernames/passwords.

Detection:  
- Filter: http.request.method == "POST"  
- Analyze HTTP payload for clear-text credentials.

**2.6 Malware Communication (Beaconing Behavior)**

Pattern: Periodic external connections to the same IP or domain.

Detection:  
- Use “Statistics > IO Graphs” to visualize timing patterns.  
- Look for repetitive intervals in outbound connections.

**3. Protocol Anomalies Detected**

|  |  |  |
| --- | --- | --- |
| Protocol | Anomaly Type | Risk |
| HTTP | Cleartext passwords | Credential theft |
| DNS | Tunneling data via queries | Data exfiltration |
| TCP | Repeated SYNs without ACKs | SYN flood |
| ARP | Conflicting MAC addresses | Spoofing / MITM |
| ICMP | Ping sweeps / scans | Reconnaissance |

**4. Mitigation Techniques**

|  |  |  |
| --- | --- | --- |
| Threat | Detection Method | Mitigation Strategy |
| ARP Spoofing | ARP filters in Wireshark | Use static ARP entries, implement DAI |
| DNS Tunneling | Monitor DNS query length/frequency | Block suspicious domains, use DNS firewall |
| SYN Flood | Check TCP handshakes | Enable SYN cookies, firewall rate limit |
| Port Scanning | Monitor port access patterns | Use IPS, limit open ports |
| Unencrypted Credentials | Inspect HTTP POST requests | Use HTTPS, enforce strong auth |
| Malware Beaconing | Track periodic outbound traffic | Block C2 IPs, use EDR tools |

**5. Recommendations**

- Always capture traffic on critical endpoints and gateway interfaces.  
- Automate detection using Wireshark display filters and Lua scripts.  
- Integrate Wireshark findings with SIEM tools for correlation.  
- Conduct periodic reviews of network logs and protocol behavior.  
- Train staff on interpreting protocol anomalies and malicious signatures.

**6. Conclusion**

Wireshark remains a cornerstone tool in identifying subtle yet dangerous network threats. By focusing on anomalies in protocol behavior and suspicious traffic patterns, administrators can proactively detect and mitigate threats such as spoofing, DoS, and data exfiltration. Combining manual inspection with automation and best practices significantly enhances network defense.

***6.Detection of Network Threats Using Zeek***



Focusing on Suspicious Patterns and Protocol Anomalies

**1. Introduction**

Zeek (formerly known as Bro) is a powerful, script-driven network monitoring tool designed to analyze traffic in real-time and generate logs for network events. Unlike packet analyzers like Wireshark, Zeek provides a higher-level understanding by interpreting protocol behavior, making it highly effective for detecting suspicious patterns and anomalies in network traffic. This report outlines key threat types identified using Zeek and strategies for mitigation.

**2. Common Detected Threats in Zeek**

**2.1 Brute Force Login Attempts**

Pattern: Multiple failed login attempts from a single IP in a short time frame.

Detection:  
- Detected in Zeek's `ssh.log`, `ftp.log`, or custom policy scripts.  
- Look for repeated failed attempts to the same account.

**2.2 DNS Tunneling**

Pattern: Excessive long domain name queries and frequent TXT record lookups.

Detection:  
- Analyze `dns.log` for domain lengths and record types.  
- Custom scripts can flag unusually long or frequent queries.

**2.3 Suspicious HTTP User-Agents or URIs**

Pattern: User-agents used by malware or requests to abnormal endpoints.

Detection:  
- Check `http.log` for rare or outdated user-agents.  
- Analyze URLs with encoded payloads or suspicious patterns.

**2.4 Port Scanning and Reconnaissance**

Pattern: Connections to a wide range of ports from the same source IP.

Detection:  
- Use `conn.log` to detect high-frequency scanning behavior.  
- Correlation of failed connections using custom Zeek scripts.

**2.5 SSL/TLS Certificate Issues**

Pattern: Self-signed certificates or mismatched domains in SSL sessions.

Detection:  
- Analyze `ssl.log` for anomalies in cert validation, issuer names, and expired certs.

**3. Protocol Anomalies Detected**

|  |  |  |
| --- | --- | --- |
| Protocol | Anomaly Type | Risk |
| SSH | Repeated login failures | Brute-force access |
| DNS | Large/frequent queries | Data tunneling |
| HTTP | Suspicious URIs/User-Agents | Malware C2 traffic |
| TCP | Port scanning | Reconnaissance |
| SSL | Self-signed/expired certs | Man-in-the-middle risk |

**4. Mitigation Techniques**

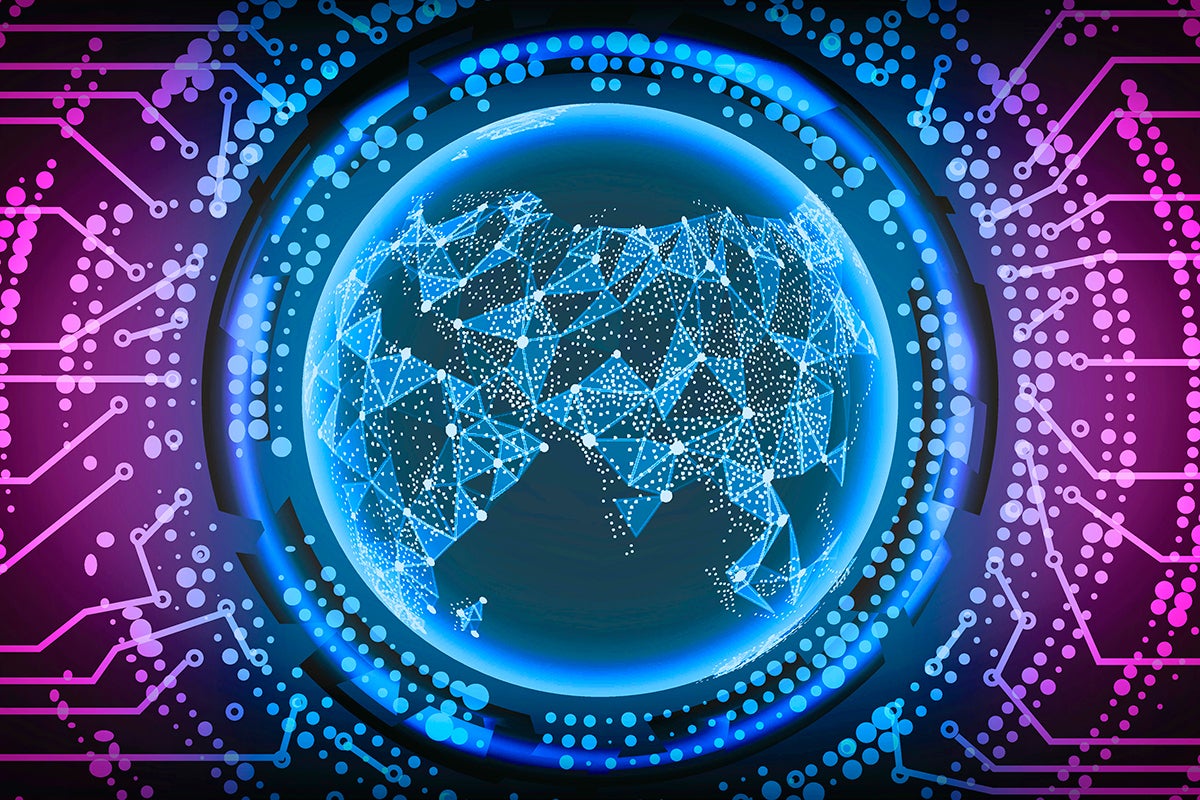
|  |  |  |
| --- | --- | --- |
| Threat | Detection Method | Mitigation Strategy |
| Brute Force Logins | Monitor ssh.log, ftp.log | Rate limiting, account lockout policies |
| DNS Tunneling | dns.log + custom scripts | Block suspicious domains, limit query length |
| HTTP Anomalies | http.log (User-Agent, URI) | Filter malformed URIs, deny known bad agents |
| Port Scanning | conn.log pattern analysis | Firewall rules, scan detection alerts |
| SSL Anomalies | ssl.log certificate checks | Block self-signed certs, enforce cert validation |

**5. Recommendations**

- Deploy Zeek at network ingress/egress points for maximum visibility.  
- Write or integrate Zeek scripts to detect repeated access attempts or data leaks.  
- Regularly analyze Zeek logs for patterns of reconnaissance, tunneling, or malicious signatures.  
- Forward Zeek logs to a SIEM (e.g., Splunk, ELK) for alerting and correlation.  
- Maintain up-to-date threat intelligence feeds for real-time IOC correlation.

**6. Conclusion**

Zeek offers a scalable and scriptable platform for detecting advanced network threats and protocol misuse. Through intelligent parsing of traffic and behavioral pattern recognition, Zeek enables proactive detection of brute-force attacks, data exfiltration attempts, and suspicious activity. Coupled with alerting and automation, Zeek strengthens the overall security posture of the organization.

7. Conclusion

Both Wireshark and Zeek serve distinct but complementary purposes in network traffic analysis. Wireshark excels at packet-level inspection and forensic analysis, whereas Zeek provides broader insights and automated event detection. When used together, they offer a comprehensive picture of network health and security.  
  
This project successfully demonstrates how combining both tools enhances traffic visibility, improves threat detection, and aids in protocol-level troubleshooting.

# 8. Recommendations



- Deploy Zeek at the network gateway to continuously monitor traffic.  
- Use Wireshark for specific incident investigation and protocol-level debugging.  
- Automate analysis scripts in Zeek for frequent threat patterns (e.g., port scanning, brute force).  
- Integrate Zeek logs with SIEM tools like Splunk or ELK stack for real-time alerts.  
- Conduct regular traffic audits and log reviews for proactive security.

# 9. References



1. Paxson, V. (1999). Bro: A System for Detecting Network Intruders in Real-Time. Computer Networks.  
2. Sanders, C. (2017). Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems. No Starch Press.  
3. Zeek Documentation. https://docs.zeek.org  
4. Wireshark Documentation. https://www.wireshark.org/docs/  
5. Scarfone, K., & Mell, P. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS), NIST.

# 10. Appendix



Appendix A: Sample Zeek HTTP Log Entry  
#fields ts uid id.orig\_h id.resp\_h method host uri status\_code user\_agent  
1680038371.432 C2kT3f2k2h 192.168.0.5 172.217.164.110 GET example.com /index.html 200 Mozilla/5.0  
  
Appendix B: Wireshark Filter Examples  
- http.request.method == "GET"  
- tcp.analysis.retransmission  
- ip.addr == 192.168.0.5  
  
Appendix C: Screenshots  
- Wireshark dashboard  
- Zeek log files  
- Protocol statistics charts