**Using Wireshark by Lana Bracken**

**1. What Did You Do?**

To explore and analyze network traffic, I used Wireshark, a packet analyzing tool. Wireshark functions by capturing network traffic directly from a selected network interface, creating a copy of each packet without interfering with live communication. The software then analyzes and dissects captured packets according to known network protocol standards such as Ethernet, IPv4, IPv6, TCP, and UDP. It presents the dissected packet information in a structured and uniform way on its user interface.

I installed Wireshark on my work laptop, accepting all default settings during setup. Upon first opening the application, I found the interface overwhelming, so I consulted instructional videos and online resources to understand basic navigation and core features. I selected my Wi-Fi network interface for capturing traffic and enabled promiscuous mode to capture all visible packets, not just those directed to my device.

For my initial packet captures, I planned two phases:

* A **baseline capture** while the network was idle, to establish a profile of normal background activity.
* A **with-events capture** where I deliberately generated traffic by interacting with various smart devices on my home network.

To systematically generate event-driven traffic, I performed the following specific actions during the **with-events capture**:

* **Google Home**: Initiated a voice query by asking, "Hey Google, what is the temperature in Canyon, TX?"
* **Smart Thermostats**: Opened the smart thermostat app on my iPhone and adjusted the target temperature on both connected thermostats.
* **Laptop**: Opened the Google Chrome browser and navigated to two websites: wtamu.edu (secured) and <http://test.unsecure.org> (deliberately unencrypted).
* **Smart TV**: Launched a video streaming application and began playing a video.
* **iPhone (Email)**: Opened the email application and refreshed the inbox to check for new messages.
* **Apple TV**: Streamed a movie trailer and used Apple AirPlay to send a photograph from my iPhone to the Apple TV.
* **Smart Vacuum (iRobot)**: Opened the iRobot app on my iPhone, started a cleaning session briefly, and then stopped it.
* **iPhone (Calls)**: Initiated both a standard cellular call and a FaceTime audio/video call.
* **Printer**: Sent a print job from both my iPhone and my laptop to the network printer.

After completing these two initial captures, I realized that identifying and isolating print job traffic — especially when initiated from mobile devices — was more challenging than anticipated. To further investigate this, I conducted two additional targeted captures:

* A **print-from-phone capture**, where I attempted to print a document from my smartphone while capturing traffic.
* A **print-from-laptop capture**, where I printed a simple "Hello World" document from my laptop to the printer.

Throughout all captures, I saved the resulting packet files for detailed post-capture analysis. I also used nmap(nmap -sn 192.168.1.0/24) to perform ping scans of my local network, identifying active devices and correlating IP addresses observed in the captures with known systems.

For analysis, I explored Wireshark's features including:

* **Protocol Hierarchies** to review overall protocol distribution.
* **Conversations and Endpoints** to study communication between devices.
* **Follow TCP Stream** to isolate specific sessions.
* **Display Filters** to target protocols or devices of interest.
* **Expert Information** to identify retransmissions, resets, and unusual behavior.
* **I/O Graphs** to visualize traffic spikes and volume trends.

This multi-stage capture approach, combined with structured interaction events and layered analysis, provided a rich foundation for understanding both normal and event-driven network activity.

**2. What are the results?**

**2.1 Overview of Networks and Protocols Identified**

In total, four packet captures were collected and analyzed:

1. **Baseline Capture** – representing idle background network traffic.
2. **With-Events Capture** – where deliberate device interactions were performed to generate traffic.
3. **Print-from-Phone Capture** – targeted capture while attempting to print from a mobile device.
4. **Print-from-Laptop Capture** – targeted capture while printing a document from a laptop.

These captures provided insight into normal background traffic, device-initiated sessions, and the network behavior of specific actions such as printing. In each capture, protocol distributions, source and destination IP addresses, and packet content summaries were examined to build an overall understanding of the network's behavior and potential vulnerabilities.

**Baseline Capture Findings**:

* **Network**: Encrypted private Wi-Fi (WPA2)
* **Top Protocols Observed**:
  + UDP (55.05%) and TCP (36.69%) were the most common transport protocols, indicating that even during idle periods, various applications and background processes are actively sending and receiving data using both connection-oriented and connectionless methods.
  + QUIC protocol traffic (6.97% of packets, 5.37% of bytes) indicated modern encrypted communications.
  + TLS traffic (7.77% of packets, 10.7% of bytes) showed encryption even during idle periods.
  + mDNS (Multicast DNS) and SSDP protocols were active, reflecting IoT device discovery, indicating that smart devices on the network are constantly announcing their presence and searching for other services.

**With-Events Capture Findings**:

* **Network**: Same private Wi-Fi network
* **Top Protocols Observed**:
  + TCP (59.79%) and UDP (31.00%) dominated, showing a shift towards more reliable, connection-oriented communication likely due to active browsing and streaming.
  + TLS traffic increased dramatically, accounting for 88.43% of total data volume, indicating a significantly higher reliance on secure communication when devices were actively being used, which is a positive security posture.
  + QUIC traffic (13.44% of packets) was associated with fast web and streaming connections.
  + HTTP traffic appeared (0.80% of packets, 3.29% of bytes) from a deliberate visit to an unencrypted test site, highlighting a potential vulnerability point where data was transmitted in plain text and could be intercepted.
  + DNS traffic (3.79% of packets) supported web lookups and device connections, demonstrating the fundamental role of name resolution in all network activity.
  + mDNS and SSDP remained active due to smart device broadcasting, continuing to expose device presence on the local network.

The table below compares the protocol activity and security observations between the baseline capture and the event-driven capture.

**Table 1: Baseline vs. Event-Driven Capture Summary**

|  |  |  |
| --- | --- | --- |
| Feature | Baseline Capture | With-Events Capture |
| Total Packets | 3,976 | 8,963 |
| Dominant Transport Protocol | UDP (55%) | TCP (60%) |
| Major Application Protocols | QUIC (7%), TLS (8%), DNS, mDNS | TLS (16% packets, 88% bytes), QUIC (13%), HTTP, DNS, mDNS |
| Encryption Usage (TLS/QUIC) | Moderate (~15% traffic volume) | Very High (~95% traffic volume) |
| IoT Discovery Protocols (mDNS/SSDP) | Present | Present and active |
| HTTP (Unsecured Web Traffic) | None observed | Present during intentional test |
| ICMP Traffic (Pings) | Minimal (<0.03%) | Minimal (<0.01%) |
| SMB/NetBIOS Traffic | Negligible | Negligible |
| Attack Evidence | None | None, but HTTP noted as a vulnerability |

These results support strengthening network security by establishing baseline traffic patterns to detect anomalies, identifying persistent IoT device communications that increase the attack surface, and highlighting unencrypted traffic as a target for future security hardening efforts.

**2.2 Summary of Packet Contents and Device Activity**

Captured packets provided detailed insight into how different devices communicate:

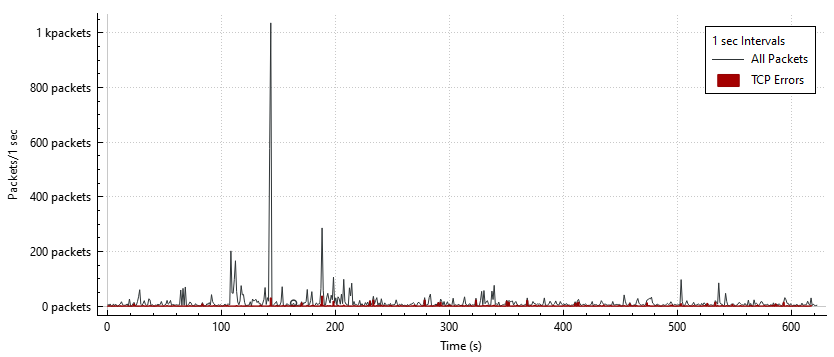
* **Google Home**: Issued DNS queries and mDNS announcements.
* **Smart Thermostats**: Communicated securely with cloud services via TLS.
* **Laptop**: Accessed HTTPS sites like <https://wtamu.edu> and a deliberately unencrypted site (<http://test.unsecure.org>) via HTTP, exposing visible GET requests.
* **Smart TV**: Initiated video streaming sessions over encrypted protocols.
* **iPhone**: Contacted Apple services and refreshed email via encrypted channels.
* **Apple TV**: Streamed content and received AirPlay transfers.
* **iRobot Smart Vacuum**: Sent brief encrypted control traffic.
* **Printer**: During the print-from-phone capture, no direct evidence of print traffic over expected ports (such as TCP 9100 or TCP 631) was observed, suggesting that the mobile printing application may have used proprietary encrypted channels or protocols not captured fully.

During the print-from-laptop capture, direct communication was visible between the laptop (192.168.1.46) and the printer (192.168.1.151) on TCP port 3910. Filtering on tcp.port == 3910 revealed approximately 70 packets, and drilling down into TCP stream 25 isolated 25 packets associated with the print session.

These results can be used to strengthen network security by mapping device behavior to establish a behavioral baseline for anomaly detection and by identifying the use of HTTP versus HTTPS to inform improvements to network encryption policies.

**2.3 Traffic Volume Analysis**

An I/O graph of the event-driven capture revealed a significant spike in packet activity between approximately 140 and 147 seconds. I filtered packets using frame.time\_relative >= 140 && frame.time\_relative <= 147. The analysis showed a burst of encrypted TLS 1.3 Application Data from a Google Cloud server (142.250.115.95) to a local device (192.168.1.46) over port 443. This reflected the start a large data transfer.



**Figure 1: I/O Graph from With-Events Capture**  
*This graph illustrates packet activity over time, with a significant spike between 140–147 seconds. This corresponds to encrypted traffic delivery from content loading.*

These results can be used to enhance network monitoring and security assessment. First, monitoring traffic spikes enables differentiation between normal high-volume activity, such as streaming, and potential data exfiltration attempts. Second, understanding which devices initiate high-volume transfers supports effective bandwidth management and risk mitigation planning.

**2.4 Expert Information Analysis**

Wireshark’s Expert Information tool was used to identify notable packet events and protocol behaviors. In the event-driven capture, the tool highlighted:

* TCP-related issues such as out-of-order segments, connection resets, and retransmissions, potentially indicating network congestion, instability, or issues with specific device communication.
* QUIC handshake decryption failures, expected due to lack of session keys, underscoring the limitations of packet inspection for encrypted protocols without the necessary decryption material.
* Routine TCP connection management (SYN, FIN, ACK sequences) as informational messages, demonstrating the normal establishment and termination of reliable network connections.

These results can be used to improve network security by tracking retransmissions and session resets to identify instability or interference, and by recognizing decryption failures as a limitation of traffic inspection without encryption keys.

During the baseline capture, warnings were primarily DNS-related, such as retransmissions and missing responses—normal in idle IoT-heavy networks. During the event-driven capture, warnings shifted toward TCP session behavior, such as retransmissions and resets, due to increased activity. No critical or malicious events were identified in either case.

These results can be used to improve network security by periodically comparing Expert Information across captures to establish normal behavior patterns and identify deviations that may indicate emerging threats.

**Comparison of Expert Information: Baseline vs Event-Driven Captures**

| **Capture** | **Warnings** | **Notes** | **Chat** |
| --- | --- | --- | --- |
| Baseline (Idle) | 5 (mostly DNS) | 6 | 8 |
| With Events (Active) | 5 (mostly TCP/QUIC) | 8 | 14 |

**Figure 2: Expert Information Summary Comparison**  
*This table compares categorized messages flagged by Wireshark's Expert Information tool during both the baseline and event-driven captures.*

**2.5 TCP Analysis Flags**

TCP anomalies such as retransmissions, duplicate acknowledgments, and partial acknowledgments were flagged using Wireshark’s tcp.analysis.flags field. This internal mechanism marks packets for analysis and correlates directly to several events listed in the Expert Information view.  
By comparing the baseline and with-evetns captures, I observed that tcp.analysis.flags highlighted retransmissions and resets in both. During the baseline capture, TCP analysis flags were minimal and mostly related to occasional device keep-alives and expected low-volume background traffic, indicating a relatively stable network during idle periods. In contrast, the with-events capture showed a noticeable increase in retransmissions, session resets, and out-of-order packets, consistent with the higher network activity caused by video streaming, browsing, and smart device interactions, suggesting that increased network load can introduce more TCP-level issues.  
This finding can be used to improve network monitoring by investigating packets flagged by TCP analysis to quickly identify congestion, packet loss, or potential malicious scanning activity.

**2.6 Evidence of Cyber Attacks**

No direct evidence of cyberattacks was observed in either the baseline or the with-events capture:

* ICMP traffic remained minimal (no ping floods), suggesting no obvious denial-of-service attempts via ICMP.
* No port scans or abnormal connection attempts occurred, indicating no immediate reconnaissance activities targeting open ports.
* No indicators of malware behavior, packet injection, or session hijacking were found, providing no direct signs of malicious software communication or attempts to manipulate network sessions.

This is an important area that requires careful consideration. As a beginner running a short capture on my own secure network during normal usage, it was unlikely that I would observe a clear or obvious cyberattack. Real attacks often involve more sophisticated, stealthy techniques that are difficult to detect in a limited time window.

However, it is useful to understand what might indicate suspicious activity when analyzing packet captures:

* I did not observe any packets that definitively indicated malicious behavior. However, certain traffic patterns could be considered suspicious and would warrant further investigation if observed.
* For example, a large number of packets originating from an unknown external IP address targeting a specific port on my machine could suggest a port scan or early stage of a denial-of-service (DoS) attack. No such activity was present in my captures.
* A sudden surge in traffic to or from an unfamiliar external IP, especially using unusual ports or protocols, might be a sign of unauthorized communications. I did not see this behavior.
* Packets with malformed headers or with unusual flags set in the TCP or IP layers could signal packet crafting attacks or exploitation attempts, but I did not identify any such anomalies.
* High volumes of unexpected broadcast or multicast traffic can sometimes suggest misconfigurations or local network attacks. In my captures, the levels of broadcast (e.g., mDNS, SSDP) were typical for an IoT-heavy home network and did not appear excessive.

Detecting cyberattacks in network traffic requires first establishing an understanding of normal baseline behavior. Deviations from that baseline - such as unexpected sources, unusual protocols, suspicious ports, or abnormal traffic volumes - can be strong indicators of potential security threats.

These results can be used to improve network security by blocking or isolating unnecessary protocols to reduce exposure and by promoting encrypted communications like HTTPS and TLS to protect sensitive information in transit.

**2.7 Analysis of Attack Surface**

The captured packets reveal several potential areas that could be part of the network attack surface:

* **Smart Devices**: Discovery protocols such as mDNS and SSDP expose device presence and available services to anyone on the local network, providing reconnaissance opportunities for attackers.
* **Unencrypted Communications**: The occasional use of HTTP traffic exposes transmitted data to possible interception by unauthorized parties.
* **Open Infrastructure Services**: Core services like DNS and DHCP are critical to network operations but could be exploited through spoofing or rogue server attacks if not properly secured.
* **Legacy Protocols**: Although minimal, the observed SMB and NetBIOS activity still represents potential risk, particularly if older versions are vulnerable to known exploits.
* **Open Ports**: While not explicitly detailed in the packet captures, open ports on devices represent potential entry points for attackers. Further analysis with another tool for deeper inspection of connection establishment packets could more clearly define this risk.
* **Multiple Protocol Usage**: The use of a wide range of protocols (TCP, UDP, HTTP, QUIC, TLS) increases the complexity of the attack surface, as each protocol has its own potential vulnerabilities.
* **External Communications**: Interactions with external websites and cloud services expand the attack surface beyond the local network. Vulnerabilities in remote servers could be exploited to compromise internal devices.
* **Wi-Fi Network Security**: The wireless network itself is a critical part of the attack surface. If the Wi-Fi password or encryption settings are weak, attackers could gain unauthorized access to internal traffic.
* **Application Layer Vulnerabilities**: Applications like web browsers, email clients, and mobile apps generate network traffic. Flaws in these applications could be exploited remotely, even if the network itself is properly secured.
* **Reliance on DNS**: While no DNS anomalies were detected, the fundamental dependence on DNS introduces potential vulnerabilities, such as DNS spoofing or cache poisoning attacks.

These results can be used to strengthen proactive network defenses by identifying attack surfaces, allowing for the segmentation of smart devices and the enforcement of secure communications to minimize exposure.

**Conclusion of Analysis**

In summary, the analysis of both baseline and event-driven captures revealed expected protocol behavior across a modern smart home network, with no direct signs of cyberattacks. Differences in traffic patterns, protocol usage, and Expert Information messages reflected the contrast between idle and active network states. By combining protocol hierarchy analysis, I/O graph insights, Expert Information examination, and TCP flag investigation, I was able to profile normal network behavior and identify areas of potential vulnerability.

While no active attacks were observed, the identification of various attack surface elements - such as persistent device discovery traffic, unencrypted communications, external service interactions, and application-layer dependencies - highlighted critical areas where network security could be strengthened. Detecting future anomalies will rely on understanding this established baseline of normal activity, monitoring for deviations, and proactively reducing exposure through segmentation and enforcement of secure protocols. These insights lay the groundwork for future proactive network security monitoring, anomaly detection, and overall network hardening strategies.

**3. What Did You Learn? (Key Takeaways)**

First and foremost, I learned that an entire semester dedicated to Wireshark and packet analysis would barely scratch the surface of what there is to know. Once I saw my initial capture, I became excited to create a second capture with intentional network events, eager to see how different device communications and services would appear in the traffic. However, I quickly realized that the sheer volume of packets generated during even a short capture was overwhelming, and that sifting through this volume to find meaningful patterns is the most challenging part of network analysis.

During my early exploration, I was curious whether activity from my wireless mouse would appear in the capture. After researching, I discovered that Bluetooth devices, such as wireless mice, operate on a separate radio protocol outside of standard TCP/IP networking and are not visible to typical Wi-Fi packet captures. This realization expanded my understanding of the different communication layers and reminded me that network analysis tools like Wireshark focus specifically on IP-based traffic, not all forms of wireless communication.

Feeling overwhelmed, I appreciated the powerful tools Wireshark provides for statistical analysis and traffic filtering. One early technique I learned was to apply simple display filters like !(arp or dns or icmp) to clear out much of the background noise and focus on traffic relevant to my investigation. I also learned to right-click any packet within a TCP conversation and select “Follow TCP Stream” - a much faster and more efficient way to view full conversations without constructing complex manual filters. I realized that knowing when to use filters strategically is key: different scenarios, such as investigating web traffic or checking for evidence of a cyberattack, require different filter techniques.

One of the most important lessons came when I tried to build a filter to identify "unknown" devices by excluding known IP addresses from my capture. What I assumed would be a simple filter quickly became a complex task when I discovered that different types of packets (ARP, DHCP, multicast, and IPv6 traffic) did not behave the way I expected. ARP operates at Layer 2, and thus my IP-based filters missed those packets. IPv6 packets weren’t excluded by IPv4-specific filters. Broadcasts and multicast communications needed additional exclusions. DHCP Discover packets used 0.0.0.0 as the source IP, bypassing simple exclusion logic. Ultimately, I learned that building accurate and meaningful filters requires a strong understanding of Ethernet frames, IP layers, and application protocols - and that my detail-oriented personality will likely require months or even years to fully master the nuances of packet analysis.

Through this experience, I also learned about the power and limitations of capture filters versus display filters. Capture filters are helpful when you know exactly what kind of traffic you want to collect, such as traffic from a specific device or using a specific protocol. However, they can be dangerous if applied prematurely, because important clues could be missed if the filter is too narrow. Display filters, by contrast, allow full captures to be carefully sifted after the fact. I now understand that both capture and display filters have an important place depending on the investigative goal.

Other technical lessons included learning how TCP flags like resets (e.g., tcp.flags.reset==1) can indicate connection problems or potential malicious behavior, and that the presence of large numbers of SYN packets could point to scanning or denial-of-service attacks. I also learned how encryption impacts visibility in Wireshark: encrypted protocols like TLS prevent detailed packet inspection unless a decryption key is available. I would be interested in learning how to upload session keys into Wireshark to decrypt captured traffic.

Another profound realization from this assignment was my naïve unawareness of how actively smart devices communicate on a network, even when they appear idle. Before capturing and analyzing my own traffic, I had assumed that devices only sent or received packets when I was actively using them. It was shocking to see that even when I believed my smart devices were “off” or not being interacted with, they were still consistently sending and receiving packets for service discovery, updates, synchronization, or internal status checks.

This experience emphasized how much happens invisibly on a digital network, and how dangerous it can be to assume that inactivity means silence at the network layer. It reinforced to me the idea that "you don't know what you don't know" - and that without tools like Wireshark and deliberate analysis, these hidden communications would have remained invisible. This new awareness fundamentally changed how I think about digital systems, network security, and device behavior moving forward.

Another surprising lesson came from analyzing the print job captures. I initially assumed that printing a simple "Hello World" document would generate a small, straightforward data transfer. However, I quickly discovered that even basic printing involved dozens of TCP packets, coordination over non-standard ports like TCP 3910, and multiple control and data exchanges between my laptop and the printer. Additionally, I was surprised that printing from my smartphone generated no easily identifiable print traffic over traditional IP printing protocols, likely because the mobile app routed data through encrypted cloud services. This experience taught me that network operations, even for seemingly simple tasks, are far more complex under the surface than they first appear - and that device type and application behavior significantly impact network visibility.

Finally, I discovered the value of hands-on practice with known malicious captures, such as those available from malware-traffic-analysis.net. These practice files, combined with guided objectives and solutions, offer an excellent way to build intuition and investigative skill in a safe environment.

Throughout this project, I quickly realized that if I attempted to list every single thing I learned or every avenue I wanted to explore, there would not be enough time or space to capture it all. Each packet, each protocol, and each anomaly opened up new questions and areas for deeper investigation. I could easily see myself going down countless rabbit holes - and thoroughly enjoying the journey. This experience showed me that packet analysis is not just a technical skill, but a fascinating exploration of how invisible digital systems truly operate.

Overall, this assignment gave me a much deeper appreciation of digital networks, the complexity of packets, the art of filtering, and the critical importance of establishing normal traffic baselines. Based on this assignment, I feel that I have developed a starting point for future threat detection by building a baseline understanding of my own network’s normal traffic patterns. In the future, these skills will enable me to better detect anomalies, understand attack surfaces, and contribute to proactive network defense strategies within any organization.

This assignment focused on my home Wi-Fi network, but the lessons learned have much broader application. In my professional role, I work for a company that provides software and technology solutions for feedyards, involving multiple networked devices such as gates, scales, dosing guns, and more. The skills I developed in packet analysis, filtering, and baseline traffic profiling could be highly valuable for troubleshooting connectivity issues, diagnosing unusual device behavior, and improving network security for these agricultural IoT systems.

In the future, I can envision using capture filters during diagnostics to isolate traffic from specific devices, identify communication failures, or detect potential security threats, thereby enhancing both operational efficiency and system protection for the organization. It’s possible that my company already uses tools like Wireshark behind the scenes, and now, with this foundational experience, I can contribute to those investigations, join in on discussions, or even help troubleshoot support tickets that involve complex network behaviors.