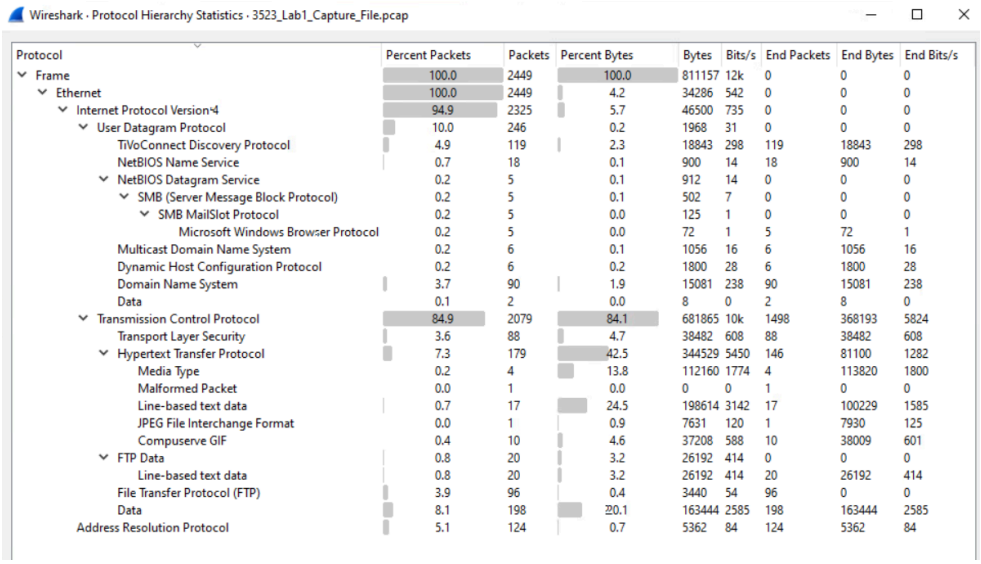
This lab focused on analyzing network traffic with Wireshark and NetworkMiner in order to tell the story of what was happening inside a captured network. The goal was not only to practice using tools but also to develop the mindset of an analyst. In class we are told that intrusion detection and incident response both depend on being able to look at raw data and turn it into a narrative that makes sense. This lab gave me the chance to do that on my own. It also showed me that the process can be frustrating at times, especially when tools do not cooperate. Even though some steps were challenging, I learned a lot about how network traffic looks, what kinds of patterns stand out, and how to decide if activity is normal or suspicious. The report that follows explains what I did, what I found, and what lessons I am taking away from the experience.

I started with Wireshark because it is the main tool for packet analysis. So looking through Wireshark the session captured lasted 8 minutes and 25 seconds on October 30, 2005. There were 2,449 packets captured and 811, 157 bytes captured. After taking a look at the protocol hierarchy, I could see that there were protocol percentages (Figure 1).

Figure 1



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	2449	100.0	811157	12k	0	0	0
Ethernet	100.0	2449	4.2	34286	542	0	0	0
Internet Protocol Version 4	94.9	2325	5.7	46500	735	0	0	0
User Datagram Protocol	10.0	246	0.2	1968	31	0	0	0
TiVoConnect Discovery Protocol	4.9	119	2.3	18843	298	119	18843	298
NetBIOS Name Service	0.7	18	0.1	900	14	18	900	14
NetBIOS Datagram Service	0.2	5	0.1	912	14	0	0	0
SMB (Server Message Block Protocol)	0.2	5	0.1	502	7	0	0	0
SMB MailSlot Protocol	0.2	5	0.0	125	1	0	0	0
Microsoft Windows Browser Protocol	0.2	5	0.0	72	1	5	72	1
Multicast Domain Name System	0.2	6	0.1	1056	16	6	1056	16
Dynamic Host Configuration Protocol	0.2	6	0.2	1800	28	6	1800	28
Domain Name System	3.7	90	1.9	15081	238	90	15081	238
Data	0.1	2	0.0	8	0	2	8	0
Transmission Control Protocol	84.9	2079	84.1	681865	10k	1498	368193	5824
Transport Layer Security	3.6	88	4.7	38482	608	88	38482	608
Hypertext Transfer Protocol	7.3	179	42.5	344529	5450	146	81100	1282
Media Type	0.2	4	13.8	112160	1774	4	113820	1800
Malformed Packet	0.0	1	0.0	0	0	1	0	0
Line-based text data	0.7	17	24.5	198614	3142	17	100229	1585
JPEG File Interchange Format	0.0	1	0.9	7631	120	1	7930	125
Compuserve GIF	0.4	10	4.6	37208	588	10	38009	601
FTP Data	0.8	20	3.2	26192	414	0	0	0
Line-based text data	0.8	20	3.2	26192	414	20	26192	414
File Transfer Protocol (FTP)	3.9	96	0.4	3440	54	96	0	0
Data	8.1	198	20.1	163444	2585	198	163444	2585
Address Resolution Protocol	5.1	124	0.7	5362	84	124	5362	84

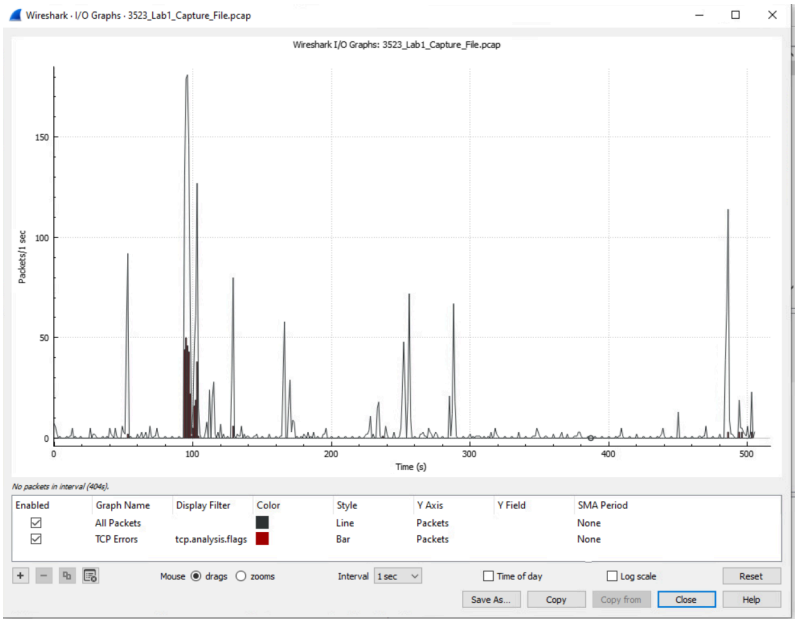
This view is helpful because it shows what protocols make up the traffic. It listed Ethernet, IP, TCP, UDP, and application protocols on top of those. Right away I could tell that most of the traffic was normal web browsing, but there were also DNS lookups and FTP sessions. Having this breakdown helps an analyst quickly see what the main activities are without going packet by packet.

The conversations view was another helpful feature. By sorting the conversations by number of packets and bytes, I could see which hosts talked the most. This made it clear that one host was at the center of most of the activity. Later I confirmed that this was the computer named KaufmanUpstairs with the IP address 172.16.1.35. Conversations view is useful because attackers usually stand out in this list if they are scanning or sending a lot of traffic. In this case, it showed me who the main person was in the capture.

The I/O graph (Figure 2) was more visual. It showed me how busy the network was at different times. At first the graph was confusing because it had spikes I could not explain, but

once I applied filters for HTTP and FTP, the graph made more sense. The HTTP spikes matched browsing sessions, and the FTP spike matched the anonymous login I later looked into. This showed me how important it is to filter data. Without filters, the graph is confusing but with them it makes it easier.

Figure 2



Another thing the I/O graph showed me is how bursts of traffic can line up with human behavior. For example, a spike in HTTP requests often matched with a user loading a web page that had many images or ads. The same is true with FTP when someone lists a directory. These bursts look very different from steady automated traffic. Recognizing that pattern helps analysts tell human activity apart from scripted attacks. It reminded me that analysis is not just technical but also about thinking how real people use networks.

After getting the big picture, I tried going into details. Using DHCP packets, I found the host name KaufmanUpstairs and the IP address 172.16.1.35. That step mattered because once I knew which host was mine, I could filter traffic just for that IP. That cut the other stuff down and let me focus on what the user on that system was doing.

Looking at browsing traffic, I saw HTTP requests to Microsoft and to a UTSA faculty site. The requests included details such as user agents, cookies, and referers. It was surprising to see how much information goes out automatically. For example, just by looking at the user agent, I could tell the operating system and browser version. That means anyone watching the traffic could figure out what software is being used and target attacks at known vulnerabilities. Even though these visits looked normal, they reminded me of the risks that come with unencrypted browsing.

The most unusual part of the capture was the FTP session. NetworkMiner showed an anonymous login with the password IEUser@. When I checked Wireshark, I saw the USER and PASS commands confirm that. The server responded with a welcome message and let the user in. The server name was linux-wlan.org.

I paid close attention to the FTP commands. The user changed directories and listed contents but never issued a RETR command. RETR is the command that downloads a file. Since it never appeared, no files were pulled down during this session. That detail made the session less alarming, but it still showed weak security. If the user had downloaded files, it would have been a red flag. The lack of transfers made this more of a yellow flag than a red one.

I also thought about why anonymous FTP was common back then. It was used to share open source software and drivers without needing user accounts. That convenience came at a cost, because it trained users to think logging in as anonymous was normal. From a modern perspective, this habit looks risky because attackers can hide in the same way. It shows how culture and habits in computing influence security, and why it takes time for best practices to replace old ways. FTP servers often allowed anonymous logins to make file distribution easier, but this practice is now considered insecure (SSH Academy, 2025).

NetworkMiner's files tab helped me confirm this. Most of the files it extracted were images, style sheets, or HTML. There was one document called IALab4(Fall05).doc, which was clearly tied to coursework. It was not an attacker tool or malware sample. That showed me the value of looking at file names and types. Malware usually stands out because it comes as .exe or .zip files, while normal browsing pulls down web content. Analysts need to know the difference so they do not panic over harmless traffic.

The local network layout was easy to figure out from DHCP options. The subnet mask was 255.255.0.0, the router was 172.16.0.1, and the DNS server was also 172.16.0.1. That told me this was a private LAN with a single gateway. Seeing the layout helped me understand where KaufmanUpstairs fit. It was just one node behind the router, talking both to local peers and to the outside world.

The story I built from all this was that a student computer was browsing normal sites and also testing access to a public FTP server. Nothing malicious, that I know of, happened in terms of file transfer, but the anonymous login was a weak spot. The network itself looked like a typical university LAN with private addresses. Putting these facts together is what event analysis is about.

Snort was the tool I struggled with the most. I tried running it on both Linux and Windows but kept hitting errors with configuration files and dynamic libraries. The snort.conf file needed changes to point to the right paths, and even after fixing those, it did not run. That was frustrating, but it taught me that IDS tools require constant upkeep. Snort is designed to catch suspicious patterns by matching traffic against rules. If the rules are not loaded correctly, or if the dynamic engines are broken, it won't work. In a real SOC, this would mean analysts lose visibility until the problem is fixed. It made me realize that tool maintenance is a big deal.

If I had gotten Snort to run correctly, it could have produced alerts that matched some of the traffic. For example, it might have flagged the anonymous FTP login, since many rulesets treat that as suspicious by default. It also could have warned about unencrypted HTTP sessions or outdated browser user agents. Even though I did not see those alerts in practice, thinking about what would have triggered them helped me imagine how an IDS adds value. It reduces guesswork and gives a second opinion on the analysis.

Even though Snort did not work for me, I read about how it should have been used. Running Snort against the capture file would trigger alerts if any traffic matched known attack signatures. That could include buffer overflows, scans, or known malware traffic. Since my

capture did not show downloads or scans, I would not expect many alerts. Still, using Snort would have been a good way to confirm what I already saw by hand. It would give useful information to what I already had.

Reflecting on this lab, I see that each tool has its role. Wireshark gave me raw visibility, NetworkMiner gave me summaries, and Snort was supposed to give me alerts. Together they form a kind of defense. This is the same idea as defense in depth. No single tool is perfect, but together they make the analyst's job easier. When one tool fails, like Snort did for me, the others can still fill in some of the gaps.

Another lesson was the importance of context. In a university, seeing anonymous FTP might just mean a student was exploring. In a corporate network, it would not be good. Analysts have to ask, is this normal for this network? If the answer is yes, then it might not be a threat. If the answer is no, then it could be an incident. This judgment is what makes analysis more than just running tools.

The lab also showed me how much data leaks in plain text. From just a few packets I saw host names, user agents, cookies, and logins. If this capture had been taken by an attacker, they would already know what systems were in use and how to target them. This reinforced why encryption matters. Today, most sites use HTTPS by default, but back in 2005 that was not the case. Seeing old captures is fascinating to me as we have better security practices now in 2025.

In the end, the suspicious activity boiled down to the FTP login. It was not a full on attack, but it was an example of weak security. The lesson is that even small things can matter. If I saw this in the real world, I would recommend blocking anonymous FTP and making sure servers were hardened. Small things can grow into bigger incidents if ignored.

Another takeaway is that doing this lab by myself gave me an appreciation for teamwork in real SOCs. In a class setting I had to push through mistakes and confusion alone, but in a job setting there would be coworkers to ask and share ideas with. Communication is part of analysis, and being able to explain what I see in plain language is just as important as clicking through tools and software. This lab gave me practice not only with technical skills but also with writing and explaining results in a clear way.

This lab also gave me a real sense of what it is like to do event analysis. It was not perfect and I made mistakes along the way, but I walked away with more confidence. I now know how to connect the dots from DHCP to browsing to FTP. I know how to use multiple tools to check my work. I also know that tools can fail, and part of the job is dealing with that. These lessons will carry over into future labs and maybe into real jobs which is great that it's a lab. It's good to practice.

In conclusion, the most difficult thing was not the analytics but how to figure out the story in a simple way. Initially the packets seemed to be all over the place but based on the combination of browsing, FTP, and local network activity, I could piece together a story people could follow. The tools are important, but the story is the one that helps others. In this lab I believe the story was about Kaufman on his computer that navigated a public FTP site but did not download files.

References

Wireshark Foundation. (2025). Wireshark Network Protocol Analyzer. https://www.wireshark.org

Netresec. (2025). NetworkMiner. https://www.netresec.com/?page=NetworkMiner

Snort Project. (2025). Snort Intrusion Detection System. https://www.snort.org

ISCS-3523 Intrusion Detection and Incident Response Course Materials, Fall 2025.

SSH Academy. (2025). *FTP Server – Beware of Security Risks*. Retrieved from SSH Academy. (2025). *FTP Server – Beware of Security Risks*.

Alexander, A. (2023, October 11). *Wireshark tutorial for beginners | Network scanning made easy* [Video]. YouTube. https://www.youtube.com/watch?v=qTaOZrDnMzQ

Motta, M. (2015, February 25). *Basics of how to use Network Miner* [Video]. YouTube. https://www.youtube.com/watch?v=vo9tK3ABw8o

Buchanan, B. (2014, January 8). *Snort with PCap files* [Video]. YouTube. https://www.youtube.com/watch?v=OSbZQG5AH2A