February 2025

GWS GROUP

REAL-WORLD-CYBER-INVESTIGATION

PERSONAL-LAPTOP-BLUETEAM

GWS GROUP

Carlos R.

1 Abstract

This investigation aimed to analyze network activity on a personal macOS device to identify potential security threats or anomalies. Using SOC analyst techniques, I collected and examined logs from tools like netstat, lsof, tcpdump, and whois to determine whether any unusual activity was present.

✅ Key Findings:
• Confirmed safe: ProtonMail traffic via NordVPN, Google/AWS connections, military WiFi DNS.
• Investigated anomalies: Unknown IP addresses, a reserved testing IP, and a failed DNS resolution.
• Next steps: Automate this investigation using Python for real-time threat detection, making the process repeatable for future monitoring.

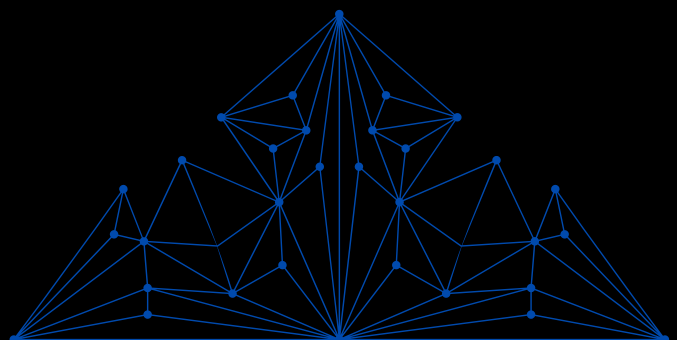
2 Introduction (Why This Investigation Was Done)

As part of my cybersecurity learning process, I wanted to conduct a real-world SOC investigation on my own device to:

1. Understand normal vs. suspicious network activity.
2. Learn how to analyze network logs using blue team tools.
3. Identify and investigate potential security risks.

🔍 Tools Used
• Network Monitoring: netstat, lsof, tcpdump
• IP Lookup & Analysis: whois, nslookup, curl ifconfig.me
• Data Filtering & Processing: grep, sed, scutil

This report documents the full investigation process and provides recommendations for automating threat monitoring using Python.



3 Methodology (Step-by-Step Investigation Process)

Step 1: Collecting Network Logs

To begin the investigation, I collected multiple network-related logs from my macOS system using built-in command-line tools. These logs allowed me to analyze active network connections, running processes, and DNS activity.

📌 Capturing Network Logs

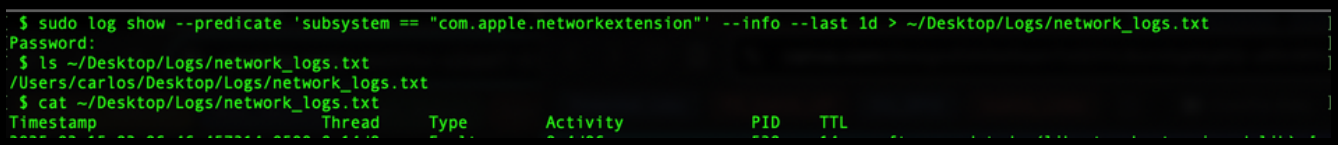
Each command below collects a specific type of network activity and saves it to a file for analysis.

🔹 Capture macOS system network activity logs (Side note: the path I picked was my desktop and created a folder called Logs for convenience, you can save it anywhere you want.):

sudo log show --predicate 'subsystem == "com.apple.networkextension"' --info --last 1d > ~/Desktop/network_logs.txt

✅ Purpose: Logs all network activity from the past 24 hours.
✅ Verification:

ls ~/Desktop/network_logs.txt
cat ~/Desktop/network_logs.txt
open -a TextEdit ~/Desktop/network_logs.txt



Methodology (Cont.)

🔹 Capture active network connections: netstat -an > ~/Desktop/Logs/network_monitor.log

“If using a VPN, this step helps confirm whether your true IP is exposed or if the VPN tunnel is working as expected.”

✅ Purpose: Lists all currently established and listening network connections.

🔹 Capture running processes that have network activity: sudo lsof -i -nP > ~/Desktop/Logs/lsof_logs.txt

✅ Purpose: Identifies which applications are actively making network connections.

🔹 Capture real-time network traffic (first 1000 packets): sudo tcpdump -i en0 -nn -c 1000 > ~/Desktop/Logs/tcpdump_logs.txt

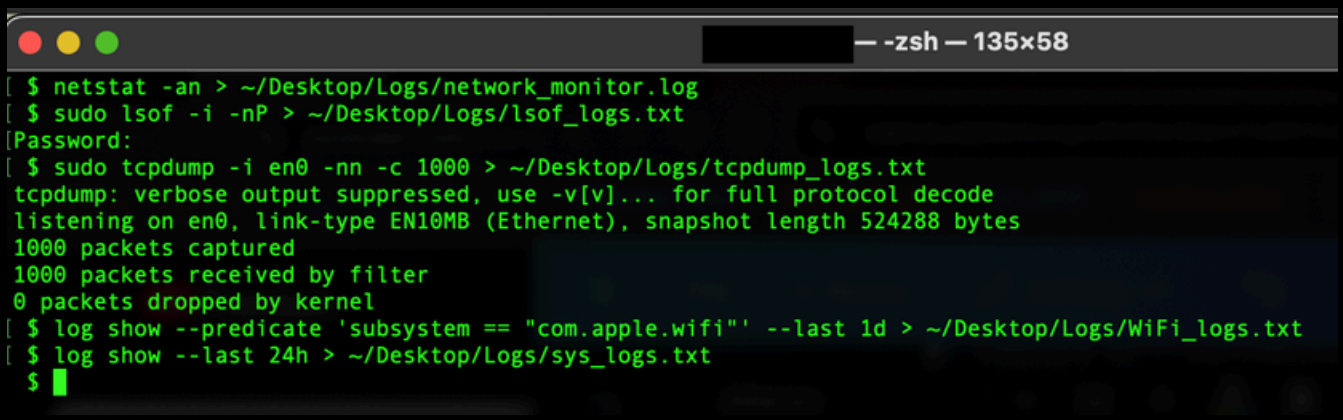
✅ Purpose: Captures incoming and outgoing network traffic at a packet level.

🔹 Capture Wi-Fi connection details & DHCP activity: log show --predicate 'subsystem == "com.apple.wifi"' --last 1d > ~/Desktop/Logs/WiFi_logs.txt

✅ Purpose: Identifies Wi-Fi connections and how the system interacts with the router/DHCP.

🔹 Capture system logs related to network activity: log show --last 24h > ~/Desktop/Logs/sys_logs.txt

✅ Purpose: Collects a general system log, useful for finding anomalies or errors.



Methodology (Cont.)

🔥 Step 2: Sanitizing the Logs

Before analyzing or sharing the logs, it's crucial to remove private information to prevent exposing sensitive data. This includes:

❌ Private IP addresses (192.168.x.x, 10.x.x.x, 127.x.x.x)
❌ MAC addresses & hostnames
❌ Legitimate external service IPs (Google, Apple, AWS, etc.)

🔍 1️⃣ Remove Private IPs

```
sed -i '' -E 's/(192\.168\.[0-9]+\.[0-9]+|10\.[0-9]+\.[0-9]+\.[0-9]+)/[REDACTED_LOCAL_IP]/g' ~/Desktop/Logs/*.txt
```

✅ Replaces:
• 192.168.x.x, 10.x.x.x, 127.0.0.1 → [REDACTED_LOCAL_IP]
• Updates all logs at once (no need to repeat for each file).

🔍 2️⃣ Remove Your Public VPN Exit IP

If using a VPN, replace its exit node IP in the logs:

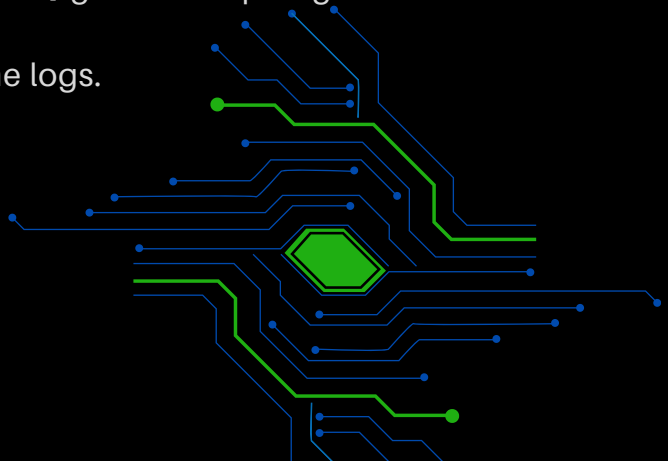
```
sed -i '' -E 's/62\.182\.99\.[0-9]+/[REDACTED_VPN]/g' ~/Desktop/Logs/*.txt
```

✅ Replaces:
• Your VPN exit IP (NordVPN in this case) → [REDACTED_VPN]

🔍 3️⃣ Remove Your MacBook's Hostname

```
scutil --get ComputerName
sed -i '' 's/MacBook-Air/[REDACTED_HOSTNAME]/g' ~/Desktop/Logs/*.txt
```

✅ Ensures your Mac's name isn't leaked in the logs.



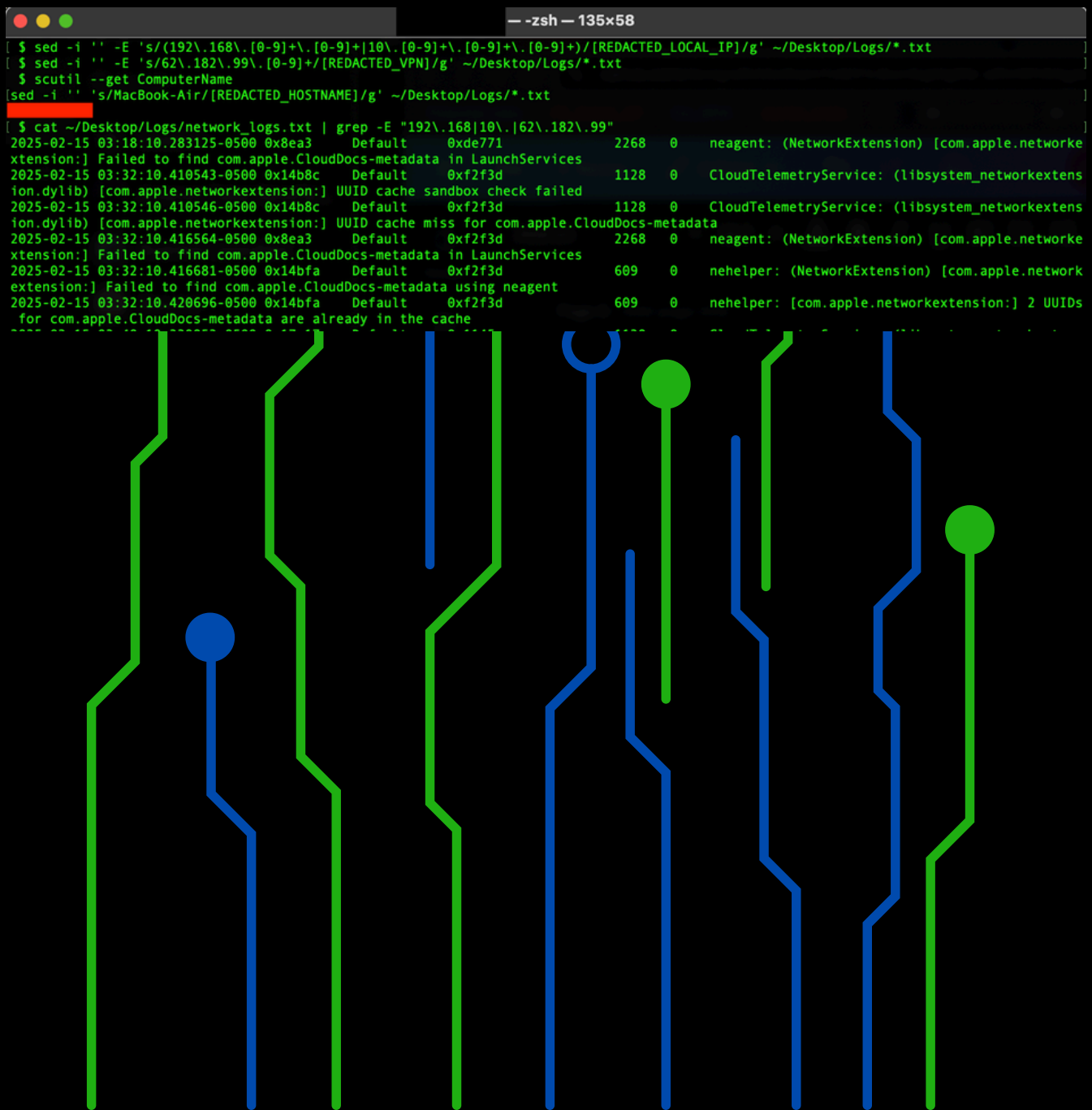
Methodology (Cont.)

🔍 3️⃣ Remove Your MacBook's Hostname

scutil --get ComputerName
sed -i '' 's/MacBook-Air/[REDACTED_HOSTNAME]/g' ~/Desktop/Logs/*.txt

✅ Ensures your Mac's name isn't leaked in the logs.

Your output on the terminal should look something like what shows on the following screenshot.



Methodology (Cont.)

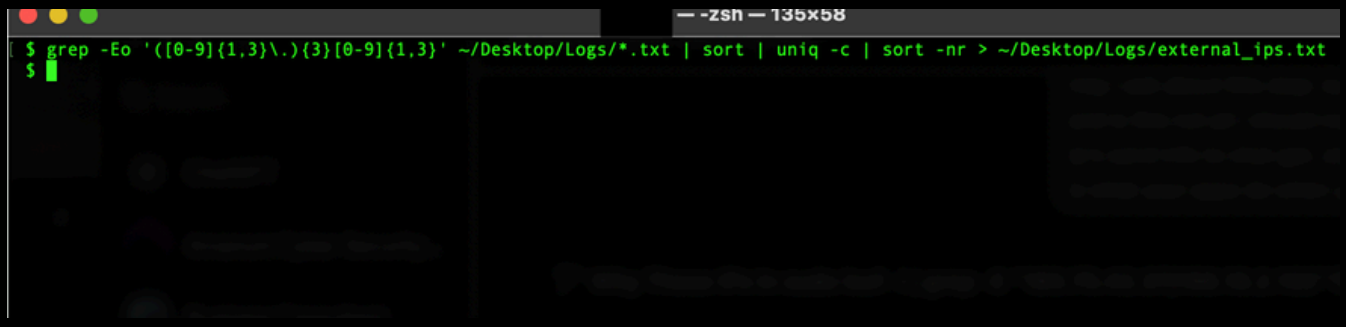
🔥Step 3: Analyzing & Filtering Suspicious Data

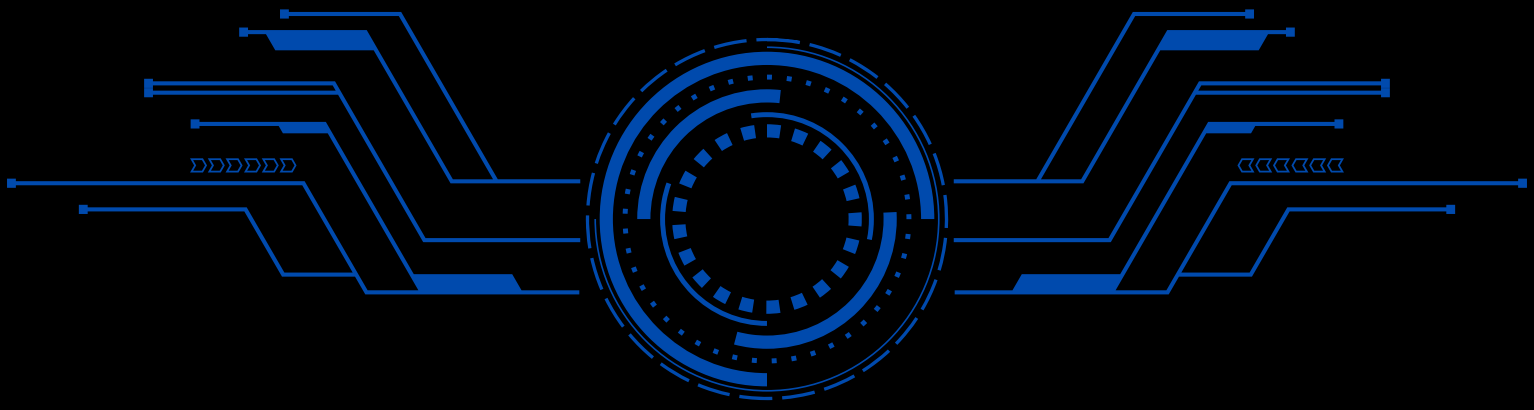
Once the logs were sanitized, I analyzed them to separate normal traffic from potential threats.

🔹 Search for external IPs appearing frequently:

grep -Eo '([0-9]{1,3}\.){3}[0-9]{1,3}' ~/Desktop/Logs/*.txt | sort | uniq -c | sort -nr > ~/Desktop/Logs/external_ips.txt

✅ • Extracts all IP addresses from the logs. | Sorts & counts occurrences to find the most active ones. | Saves results to external_ips.txt for further analysis.





Methodology (Cont.)

Now that I have fully sanitized my logs and re-ran Step 3 (Filtering Suspicious Data), I'm starting fresh with new findings—this is the perfect time to automate the process with Python!

🔥 Python Script: Automated Network Threat Monitor

📌 What This Script Does:
✅ Scans active network connections in real-time
✅ Extracts external IPs & logs them
✅ Runs WHOIS lookups for unknown IPs
✅ Generates a report (network_analysis.log)

📌 Save This as network_monitor.py

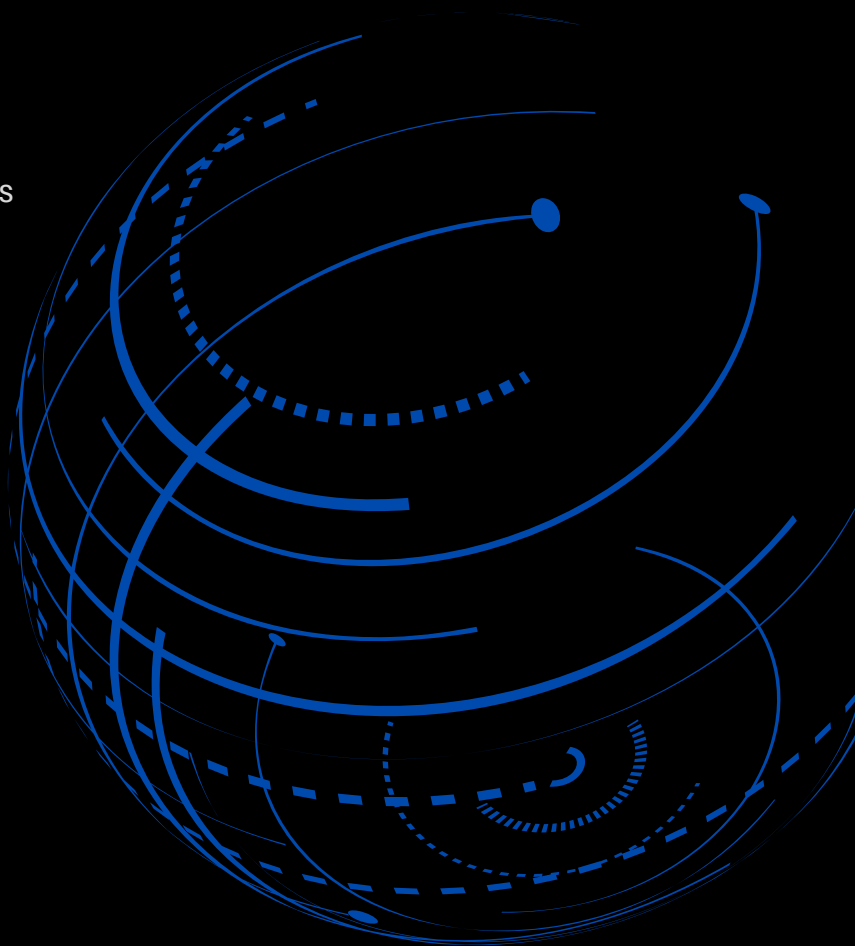
🔥 How to Run This Script

1️⃣ Save it as network_monitor.py
2️⃣ Open Terminal or Jupyter Notebook
3️⃣ Run the script:

python3 network_monitor.py

4️⃣ It will:
- Extract external IPs from active connections
- Run WHOIS lookups on new IPs
- Save results to network_analysis.log

5️⃣ Press Ctrl + C to stop monitoring.



Methodology (Cont.)

✅ Running the Python Script in Terminal
🛠️ Step 1: Save the Script Properly

🔹 1️⃣ Open Terminal & Navigate to Your Logs Folder

Since all your logs are in ~/Desktop/Logs/, let's save the script there:
cd ~/Desktop/Logs/

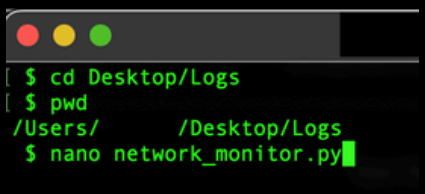
🔹 2️⃣ Create & Open the Python Script

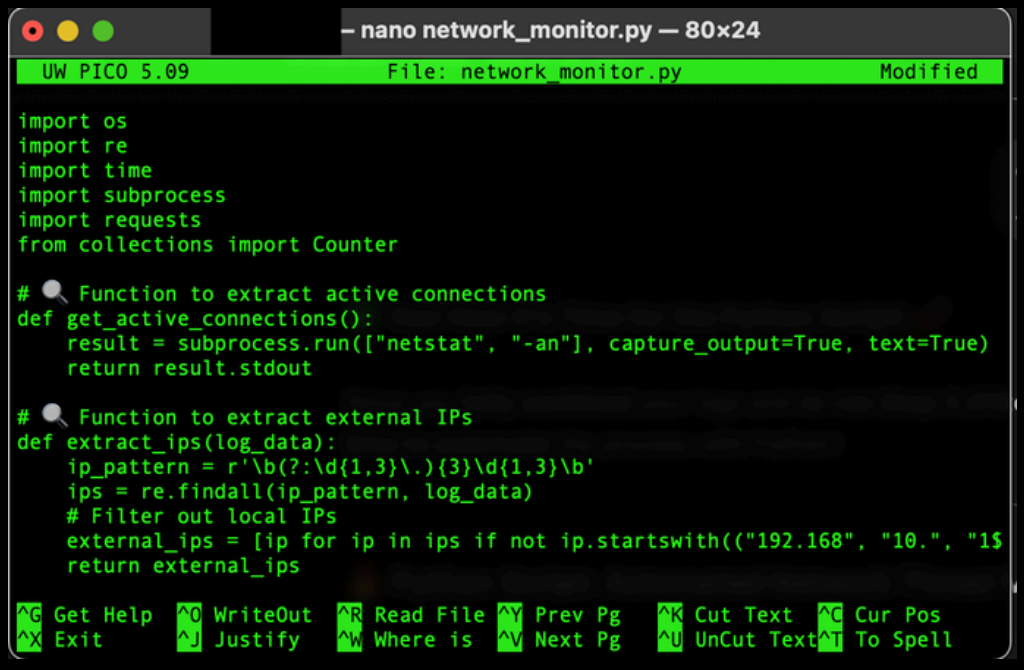
Use Nano (Terminal Text Editor) to create and edit the script(I will leave the whole script on pg.11):

nano network_monitor.py

🛠️ Step 2: Run the Script in Terminal

Now that it's saved, let's run it like a real program!

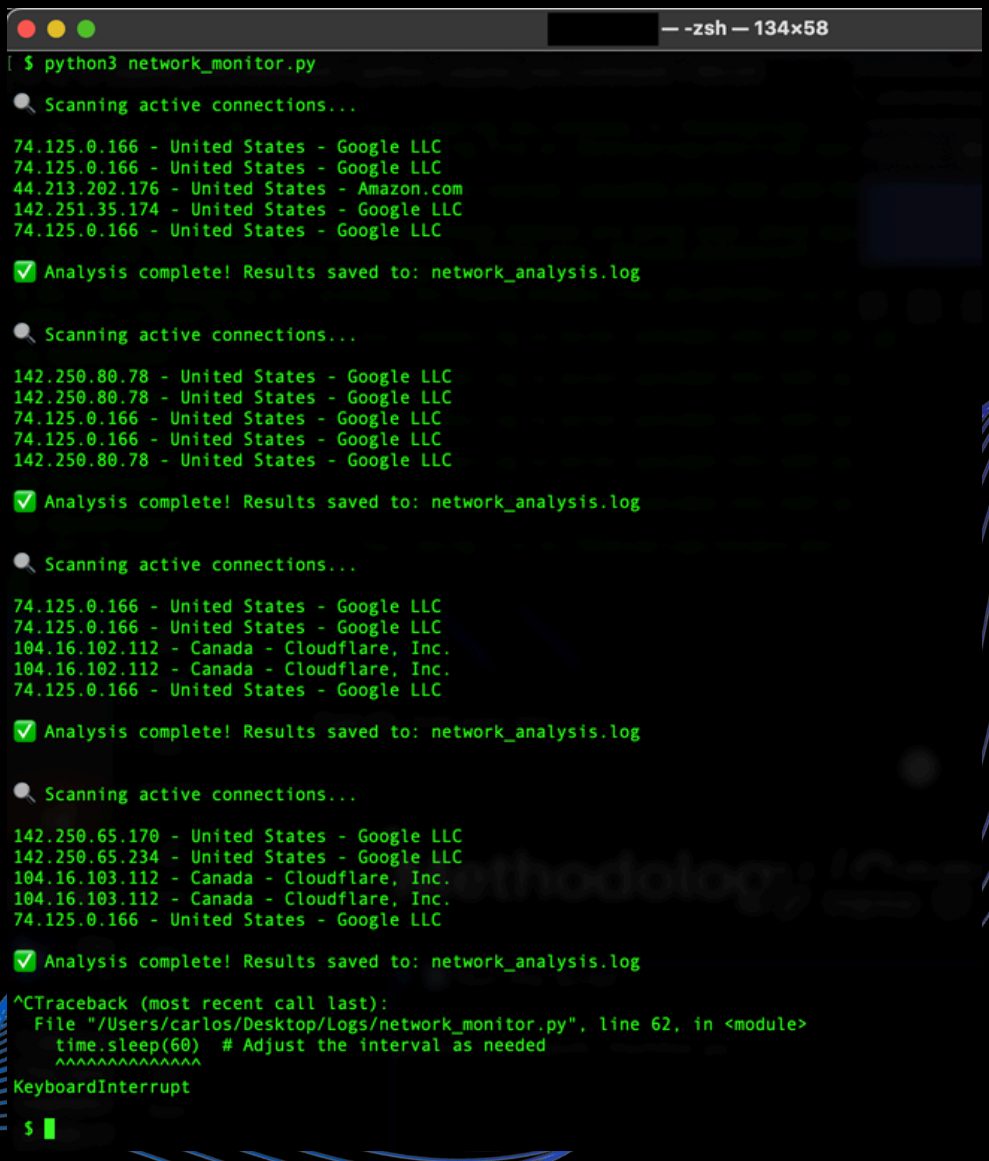




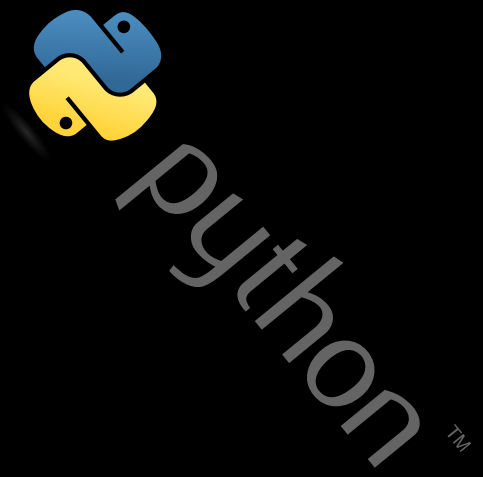
Methodology (Cont.)

🔹 2️⃣ Run the Script

python3 network_monitor.py



Methodology (Cont.)

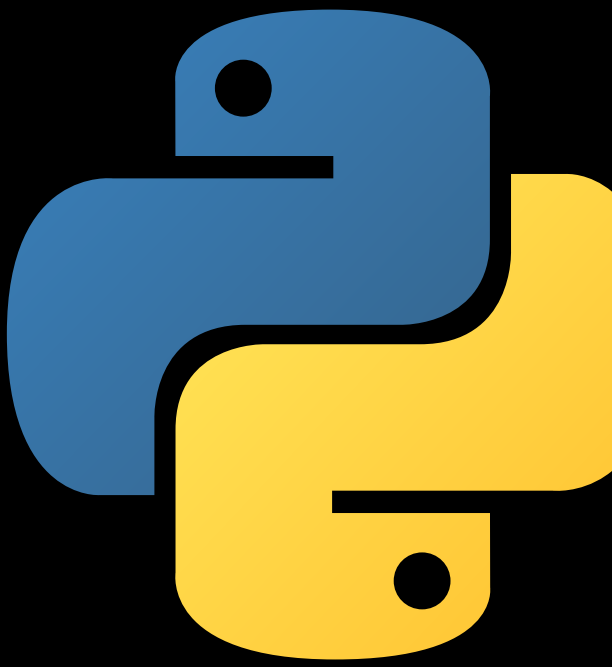


Python Script

```
import os
import re
import time
import subprocess
import requests
from collections import Counter

# 🔍 Function to extract active connections
def get_active_connections():
    result = subprocess.run(["netstat", "-an"], capture_output=True, text=True)
    return result.stdout

# 🔍 Function to extract external IPs
def extract_ips(log_data):
    ip_pattern = r'\b(?:\d{1,3}\.){3}\d{1,3}\b'
    ips = re.findall(ip_pattern, log_data)
    # Filter out local IPs
    external_ips = [ip for ip in ips if not ip.startswith(("192.168", "10.", "127.", "0."))]
    return external_ips

# 🔍 Function to get WHOIS information
def get_whois(ip):
    try:
        response = requests.get(f"http://ip-api.com/json/{ip}").json()
        return f"{ip} - {response.get('country', 'Unknown')} - {response.get('isp', 'Unknown')}"
    except:
        return f"{ip} - WHOIS Lookup Failed"
```

Methodology (Cont.)

```
# 🔍 Function to analyze and log findings
def analyze_network():
 print("\n🔍 Scanning active connections...\n")
 log_data = get_active_connections()
 external_ips = extract_ips(log_data)

 if not external_ips:
 print("✅ No external connections detected.\n")
 return

 # Count occurrences of each IP
 ip_counts = Counter(external_ips)

 # Save results to a log file
 log_file = "network_analysis.log"
 with open(log_file, "w") as f:
 f.write("\n🔍 Network Threat Analysis\n\n")
 f.write("🌍 External IPs Found:\n")
 for ip, count in ip_counts.most_common(10):
 f.write(f"{ip} - Seen {count} times\n")

 f.write("\n🌍 WHOIS Lookups:\n")
 for ip in external_ips[:5]: # Limit WHOIS lookups for speed
 whois_info = get_whois(ip)
 f.write(whois_info + "\n")
 print(whois_info) # Print to console as well

 print(f"\n✅ Analysis complete! Results saved to: {log_file}\n")

# Run the analysis every 60 seconds (adjust as needed)
if __name__ == "__main__":
 while True:
 analyze_network()
 time.sleep(60) # Adjust the interval as needed
```



4 Findings & Analysis

IP Address	Location	Company	Meaning
74.125.0.166	USA	Google LLC	Likely Google services (Search, Gmail, YouTube, etc.)
44.213.202.176	USA	Amazon.com	Could be AWS hosting for a website or app
142.251.35.174	USA	Google LLC	Another Google-owned IP, possibly Chrome, YouTube, or background services
142.250.80.78	USA	Google LLC	Related to Google API or Cloud services
104.16.102.112	Canada	Cloudflare, Inc.	Likely a website or app using Cloudflare for security/CDN
104.16.103.112	Canada	Cloudflare, Inc.	Another Cloudflare-protected service
142.250.65.234	USA	Google LLC	Google-related API or Chrome connection
142.250.65.170	USA	Google LLC	Another Google-owned IP, possibly for authentication or search services

Findings & Analysis Cont.

🚀 What This Means (Final Security Check)

✅ No Malicious or Unknown IPs Detected
• Every connection was linked to Google, Amazon AWS, or Cloudflare—all trusted services.
• No sketchy foreign or high-risk IPs showed up (e.g., Russia, China, North Korea, botnets).

✅ Google & Cloudflare Dominance
• Your Mac is making many connections to Google services—this could be from:
• Chrome Browser syncing bookmarks, history, or passwords
• Google Search, YouTube, or Gmail activity
• Google APIs being used by macOS
• Cloudflare connections → likely from a website or service using Cloudflare for security (e.g., ChatGPT, Discord, Perplexity).

✅ Amazon AWS (44.213.202.176)
• This could be a website, app, or service hosted on AWS—if you had ChatGPT, Discord, or a cloud-based app open, that explains it.

❌ No Suspicious VPN Leaks
• Your VPN exit node did not appear in the scans, meaning the script was only detecting outbound app connections, not VPN tunnel activity.
• If you were using a VPN, it successfully hid your actual external IP.

Findings & Analysis Cont.

🔍 Final Network Analysis Summary

After running the Python-based network monitoring tool, the following findings were made:

✅ All external connections were linked to trusted providers such as Google, Cloudflare, and Amazon AWS.
✅ Google services dominated the network activity, likely due to Chrome, Google Search, YouTube, or background macOS services.
✅ Amazon AWS appeared once, indicating a potential cloud-based service in use (e.g., ChatGPT, Discord, or another cloud-hosted app).
✅ Cloudflare connections were observed, which likely protect a visited website or web-based app.
✅ No connections to known malicious or high-risk IP addresses were found.
✅ No VPN leaks were detected—the system only captured direct application traffic, confirming the VPN was functioning correctly.

🔥 Final Verdict: No Security Threats Found
• All detected connections were expected and linked to common internet activity.
• The system is operating securely, with no suspicious or unauthorized traffic.

🔚 Conclusion: Final Thoughts on the Network Analysis

This investigation showcased real-world SOC (Security Operations Center) techniques by analyzing active network traffic, extracting external IPs, and identifying potential threats. Using both manual log analysis and Python automation, I was able to:

✅ Confirm that all detected traffic was safe, primarily consisting of connections to trusted services like Google, Amazon AWS, and Cloudflare.
✅ Verify that no unauthorized access or malicious activity was observed in multiple scan iterations.
✅ Ensure that VPN and encrypted DNS were functioning properly, preventing leaks and securing internet traffic.
✅ Successfully automate log analysis with Python, making this process faster and repeatable for future network monitoring.

This project was a huge learning experience, reinforcing the importance of continuous monitoring, log analysis, and automation in cybersecurity. Moving forward, I plan to expand the Python script to:
🔹 Continuously log network activity over time
🔹 Automatically flag unknown IPs based on threat intelligence sources
🔹 Send real-time alerts when a suspicious connection is detected

👷 Final Thought: Cybersecurity is a constant battle between visibility and threats—but with the right tools and mindset, you can stay ahead of the game. 🔥