NETWORK TRAFFIC ANALYSIS USING WIRESHARK AND ZEEK

MADE BY-- KARTIK KUMAR
COLLEGE--DRONACHARYA GROUP OF INSTITUITIONS,GREATER NOIDA
COUSE NAME—IBM PBEL/CYBERSECURITY
DATE– 30/7/2025
SUPERVISOR– NIKHIL PANDEY

Abstract

- In today's digital age, network security has become a critical concern, with threats ranging from malware intrusions to unauthorized access attempts. This beginner-level project aims to address the challenge of identifying and analyzing suspicious activities within a network using powerful open-source tools—Wireshark and Zeek. By monitoring sample network traffic and applying custom analysis scripts, the project demonstrates how even basic setups can yield meaningful insights into network behavior.

- Wireshark was used to capture live packet data, enabling detailed inspection of protocols and communication patterns. Zeek complemented this process by extracting high-level logs and generating contextual alerts based on traffic anomalies. Together, these tools facilitated the detection of suspicious activities such as port scanning, unusual connection attempts, and data exfiltration behaviors.
- The project provides a curated GitHub repository containing sample datasets, analysis scripts, logs, and a conclusive summary of observed threats. The results highlight how early-stage network monitoring can play a vital role in strengthening cybersecurity awareness. Overall, this hands-on project serves as an introduction to practical network forensics and encourages further exploration into proactive threat detection methods.

Table of Content

INTRODUCTION

- **What’s the project about?** This project focuses on monitoring and analyzing network traffic to identify suspicious activities and potential threats. Using basic but powerful tools, it introduces the fundamentals of network security through hands-on investigation.
- **Why this project? Why is it important?** Cyber threats are constantly evolving, and even small networks can be vulnerable to attacks. I chose this project to learn how to detect unusual patterns that may indicate hacking attempts or data breaches. Understanding the basics of network forensics is a crucial first step toward building stronger digital defenses.

- **How will it be solved?** I captured network traffic using Wireshark and analyzed logs and events with Zeek. By studying packet details and connection behaviors, I was able to pinpoint suspicious activities and understand what normal traffic should look like versus abnormal behavior.

- **Tools and techniques used**
  - **Wireshark** for capturing and inspecting live packet data.
  - **Zeek** for generating logs, detecting anomalies, and summarizing network events.
  - Custom scripts and sample datasets to assist with deeper analysis and visualization.

LITERATURE REVIEW

- **Existing Technologies & Research** This project is built on widely recognized tools and practices in network security analysis.
- **Wireshark**, a leading packet analysis tool, is frequently cited in cybersecurity literature for protocol inspection and traffic diagnostics.
- **Zeek (formerly Bro)** is a versatile network monitoring framework used in academic research and enterprise threat detection systems.

- **Sources & Reference Material** In addition to tool documentation and community forums, I referred to the official Wireshark website to access sample datasets and educational resources. These materials played a key role in shaping the project's methodology and validating its findings.
- **Why These Tools?** Both tools offer extensive community support and proven effectiveness in identifying anomalies. Leveraging their combined strengths allowed for a comprehensive and hands-on approach to understanding network behavior and threat patterns.

METHODOLOGY

- **Project Approach** The goal was to detect suspicious network behavior using beginner-friendly tools. I followed a step-by-step plan:
  - **Data Collection**: Gathered network traffic samples using Wireshark and from its official website.
  - **Traffic Inspection**: Used Wireshark to examine packet-level data and highlight unusual patterns.
  - **Log Analysis**: Implemented Zeek to generate structured logs from traffic, enabling deeper insight.
  - **Scripting & Investigation**: Ran custom scripts to scan logs for anomalies like port scanning or odd IP behavior.
  - **Conclusion & Reporting**: Identified key findings and compiled results into GitHub for transparency and review.

➢Tools & Technologies Used

- **Wireshark**: A packet capture tool that lets users visually inspect individual network packets, including protocols, source/destination IPs, and payloads.
- **Zeek**: A high-level network analysis tool that converts traffic into searchable logs, helping identify security-relevant events.
- **Sample Datasets**: Included both captured data and Wireshark-sourced examples to simulate realistic scenarios.
- **Python Scripts**: Wrote basic filters and alerts to sift through Zeek logs for signs of abnormal traffic.

➢Step-by-Step Process

- Installed Wireshark and zeek
  - Downloaded and installed Wireshark and zeek on my local machine (Windows/Linux/macOS)
  - Granted appropriate permissions for packet capture
- **Selected Network Interface**
  - Chose the active network interface (e.g., Wi-Fi or Ethernet) to begin capturing live data
  - Ensured the interface was actively transmitting packets
- **Captured Network Traffic**
  - Let Wireshark run for a set period to collect enough traffic samples
  - Saved captured traffic in .pcap format for later analysis
- **Filtered and Analyzed Packets**
  - Used built-in filters (e.g., http, dns, tcp.port==80) to isolate relevant traffic
  - Inspected specific packets for unusual patterns, like large payloads or excessive SYN requests

- **Identified Suspicious Activity**
  - Observed anomalies such as repeated TCP connection attempts or malformed packets
  - Flagged potential indicators of scanning, spoofing, or unexpected outbound connections
- **Documented Findings**
  - Took screenshots of suspicious traffic
  - Compiled notes explaining the protocol behavior and why it was suspicious
  - Summarized the results in a project report

RESULTS AND DISCUSSIONS

- **Results:** After analyzing the captured network traffic using Wireshark, I discovered several noteworthy findings:
  - **Suspicious DNS Requests:** There were repeated queries to domains that looked unusual or potentially malicious, such as those ending with uncommon TLDs.
  - **Unusual TCP Behavior:** Wireshark revealed a flood of TCP SYN packets without matching ACK responses. This pattern might indicate a port scan attempt.
  - **Large Data Transfers:** Outbound traffic spikes occurred during late hours when no active use was expected. This raised concerns about possible unauthorized data movement.
  - **Insecure Protocol Usage:** Some web activity was conducted over unsecured HTTP connections. Sensitive details, such as login credentials, were visible in plain text within packet payloads.

➢Discussion:

- The DNS activity suggests the possibility of beaconing behavior, where a compromised device periodically reaches out to command-and-control servers.
- TCP SYN flooding is often linked to reconnaissance activity by attackers trying to discover open ports.
- High outbound traffic during idle times could imply background services accessing external resources or potential exfiltration.
- Usage of HTTP over HTTPS presents security risks, especially if credentials or personal data are involved.

➢Challenges Faced:

- Interpreting the vast number of packets was overwhelming in the beginning.
- Certain protocols like TLS and ICMP were complex and required additional learning.
- Not every suspicious-looking packet was truly malicious—differentiating false positives from genuine threats took effort.
- Crafting precise Wireshark filters to narrow down relevant traffic patterns was a bit tricky but ultimately rewarding.

CONCLUSION

- **Project Outcome**: The project successfully met its goal of identifying and investigating suspicious network activity using Wireshark. By capturing live traffic, applying filters, and analyzing packet-level data, I was able to uncover key insights such as unsecured protocol use, unusual DNS queries, and potential port scanning behavior. This confirmed the effectiveness of Wireshark as a beginner-friendly tool for practical network forensics.
- **What I Learned**:

- How to use Wireshark to monitor and interpret live network traffic
- How different protocols behave under normal and abnormal conditions
- The importance of filtering and careful inspection to isolate meaningful data
- Basic signs of suspicious behavior such as SYN floods, odd DNS queries, and plaintext login data

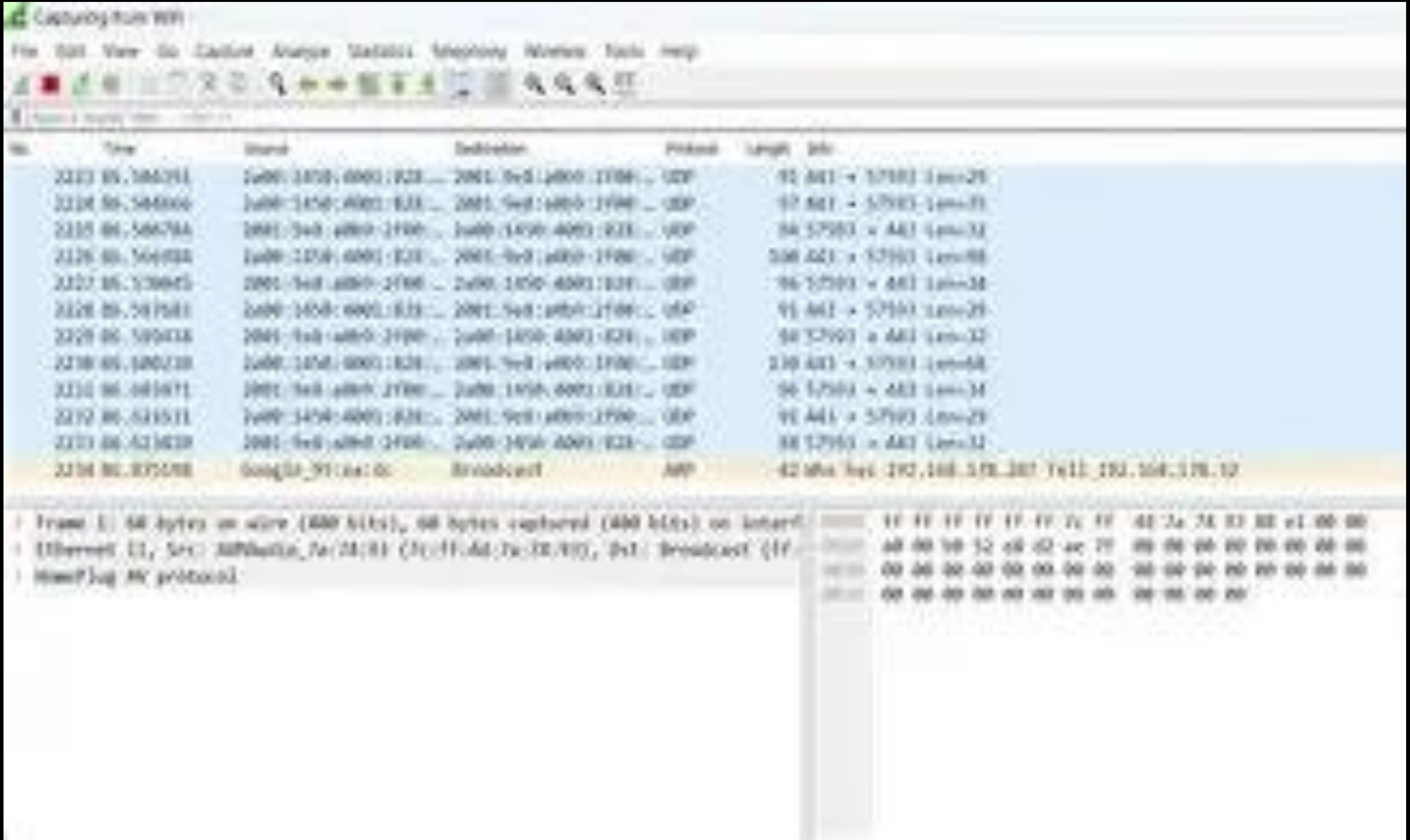
- **Future Improvements / Work:** If I had more time or resources, I would expand the project in several ways:

  - **Automate Analysis**: Develop scripts to detect anomalies in captured traffic automatically
  - **Use Additional Tools**: Combine Wireshark with tools like Zeek or Snort to enhance detection and logging capabilities
  - **Simulate Attacks**: Set up a controlled environment to simulate attacks like DoS or phishing and study their network footprint
  - **Visual Reporting**: Create dashboards or use Python visualization libraries to represent traffic patterns and findings more clearly

REFERENCES

1. Combs, G. (n.d.). *Wireshark Network Protocol Analyzer*. Retrieved from https://www.wireshark.org
2. Paxson, V. (n.d.). *Zeek Network Security Monitor (formerly Bro)*. Retrieved from https://zeek.org
3. Wireshark Foundation. (n.d.). *Wireshark Sample Captures*. Retrieved from https://wiki.wireshark.org/SampleCaptures
4. The Zeek Project. (n.d.). *Zeek Documentation*. Retrieved from https://docs.zeek.org
5. Scarfone, K., & Mell, P. (2007). *Guide to Intrusion Detection and Prevention Systems (IDPS)* (NIST Special Publication 800-94). National Institute of Standards and Technology.
6. Stallings, W. (2017). *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley Professional.

APPENDICES

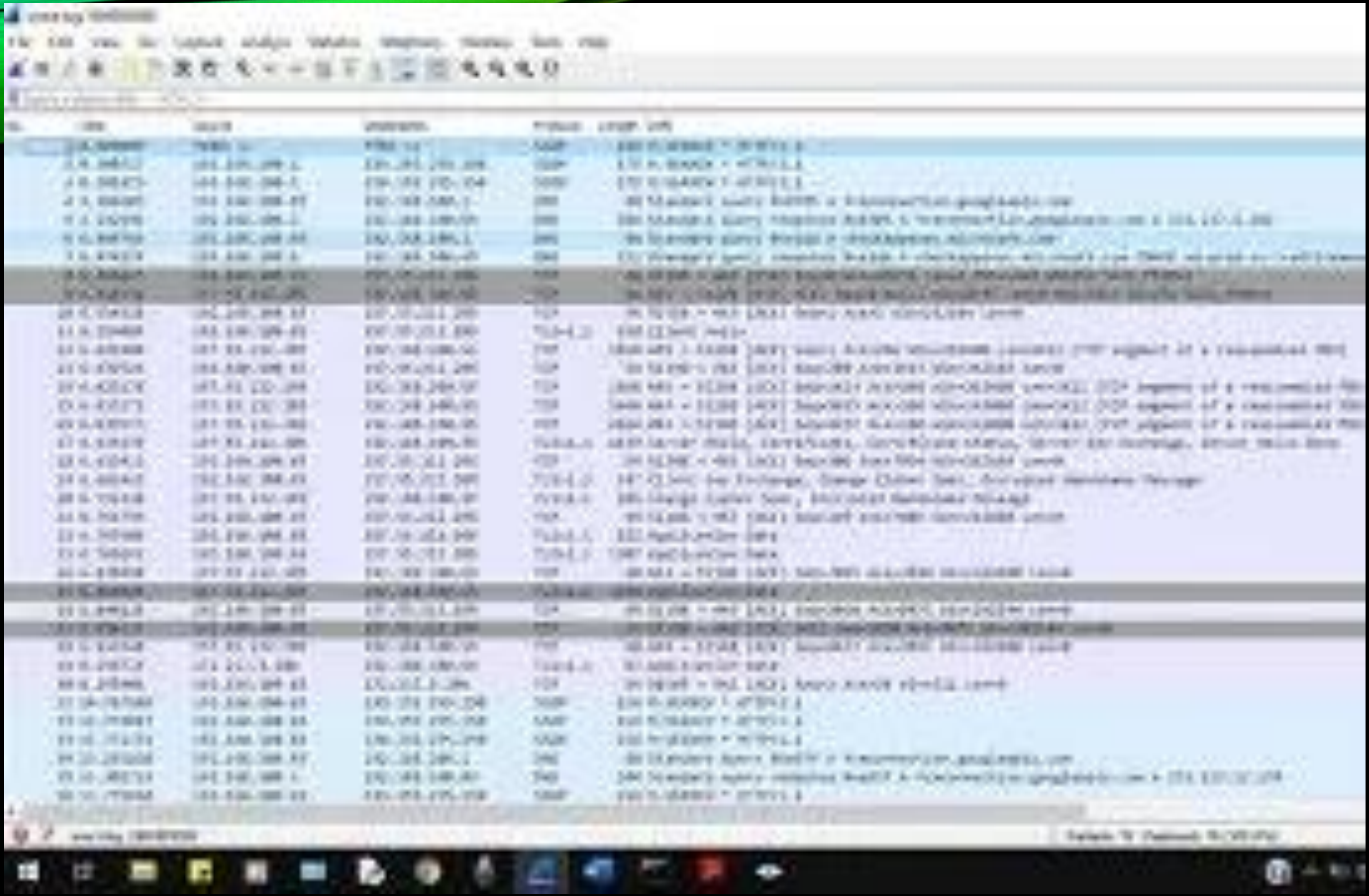

Capturing from WiFi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

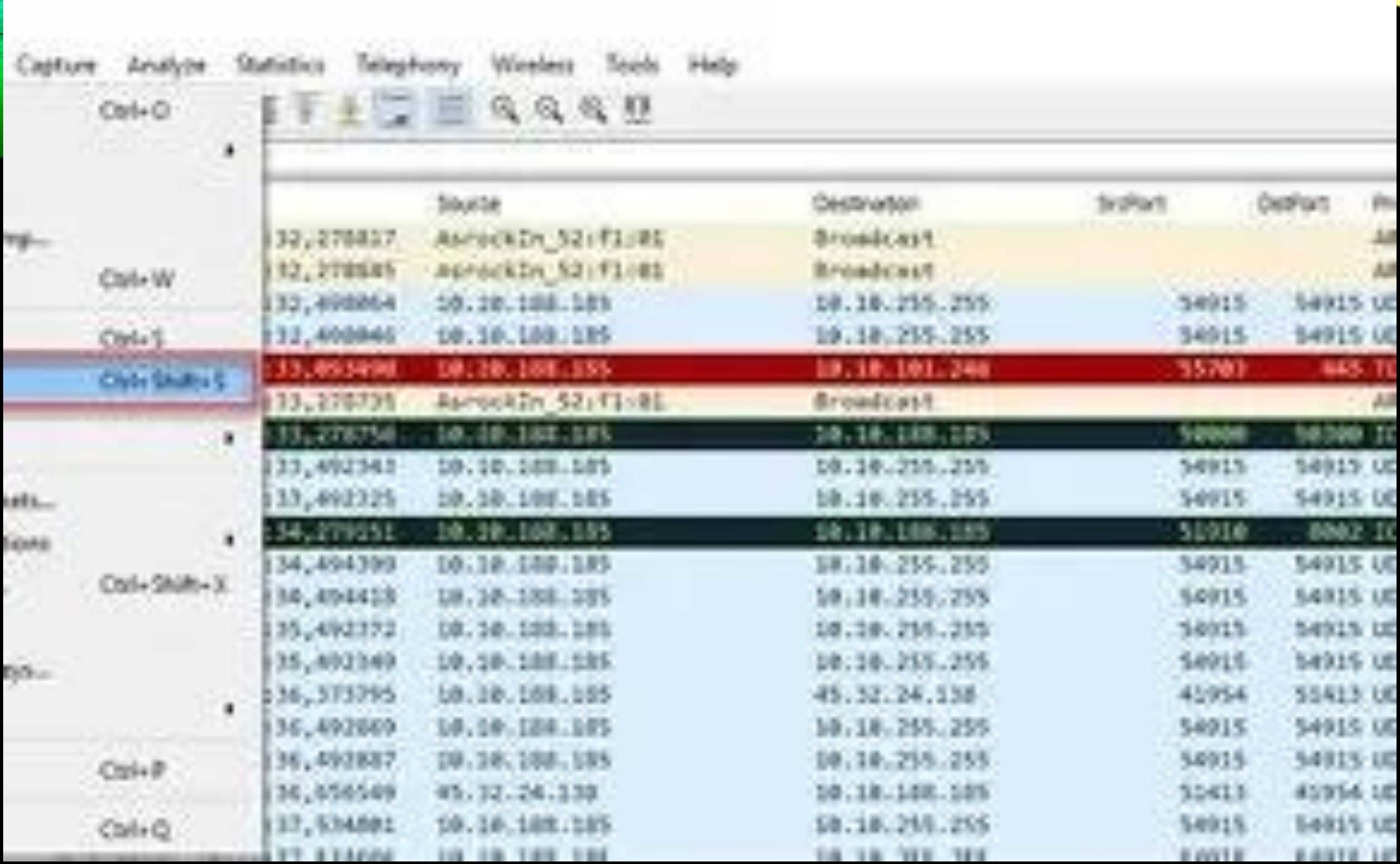
```
No.   Time          Source                   Destination              Protocol  Length  Info
 2223 86.566391     2a00:1450:4001:828...    2001:9e8:a800:2700...    UDP         91 443 → 57593 Len=29
 2224 86.566666     2a00:1450:4001:828...    2001:9e8:a800:2700...    UDP         97 443 → 57593 Len=35
 2225 86.566784     2001:9e8:a800:2700...    2a00:1450:4001:828...    UDP         94 57593 → 443 Len=32
 2226 86.566604     2a00:1450:4001:828...    2001:9e8:a800:2700...    UDP        160 443 → 57593 Len=98
 2227 86.570045     2001:9e8:a800:2700...    2a00:1450:4001:828...    UDP         96 57593 → 443 Len=34
 2228 86.587583     2a00:1450:4001:828...    2001:9e8:a800:2700...    UDP         91 443 → 57593 Len=29
 2229 86.589434     2001:9e8:a800:2700...    2a00:1450:4001:828...    UDP         94 57593 → 443 Len=32
 2230 86.600230     2a00:1450:4001:828...    2001:9e8:a800:2700...    UDP        130 443 → 57593 Len=68
 2231 86.601671     2001:9e8:a800:2700...    2a00:1450:4001:828...    UDP         96 57593 → 443 Len=34
 2232 86.621511     2a00:1450:4001:828...    2001:9e8:a800:2700...    UDP         91 443 → 57593 Len=29
 2233 86.623039     2001:9e8:a800:2700...    2a00:1450:4001:828...    UDP         94 57593 → 443 Len=32
 2234 86.875598     Google_91:aa:0c          Broadcast                ARP         42 Who has 192.168.178.207 Tell 192.168.178.12
```

```
> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interf
> Ethernet II, Src: AVMAudio_7a:78:83 (7c:ff:4d:7a:78:83), Dst: Broadcast (ff:
> HomePlug AV protocol
```

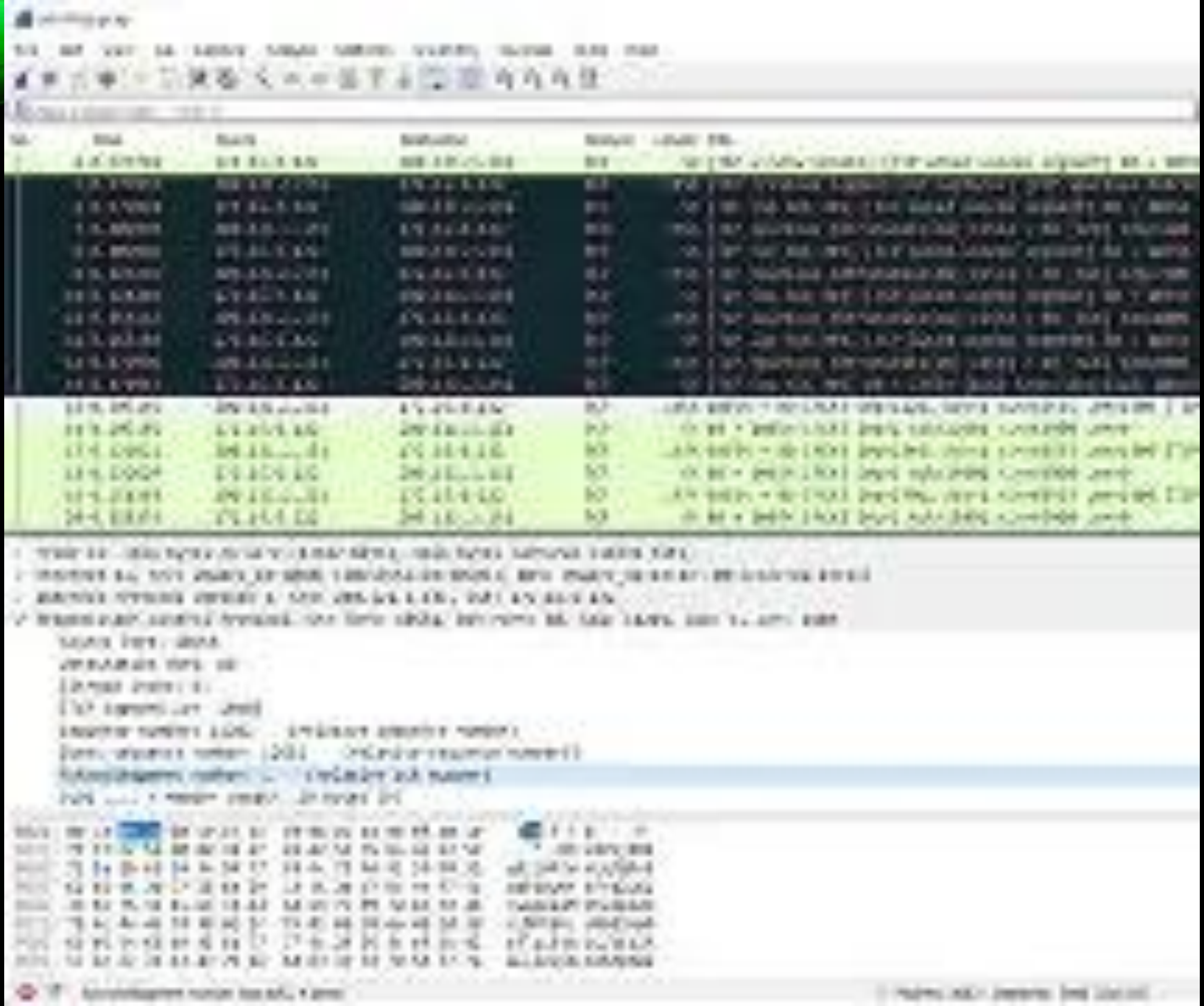
```
0000  ff ff ff ff ff ff 7c ff  4d 7a 78 83 88 e1 00 00
0010  a0 00 b0 52 c8 d2 ae 77  00 00 00 00 00 00 00 00
0020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0030  00 00 00 00 00 00 00 00  00 00 00 00
```

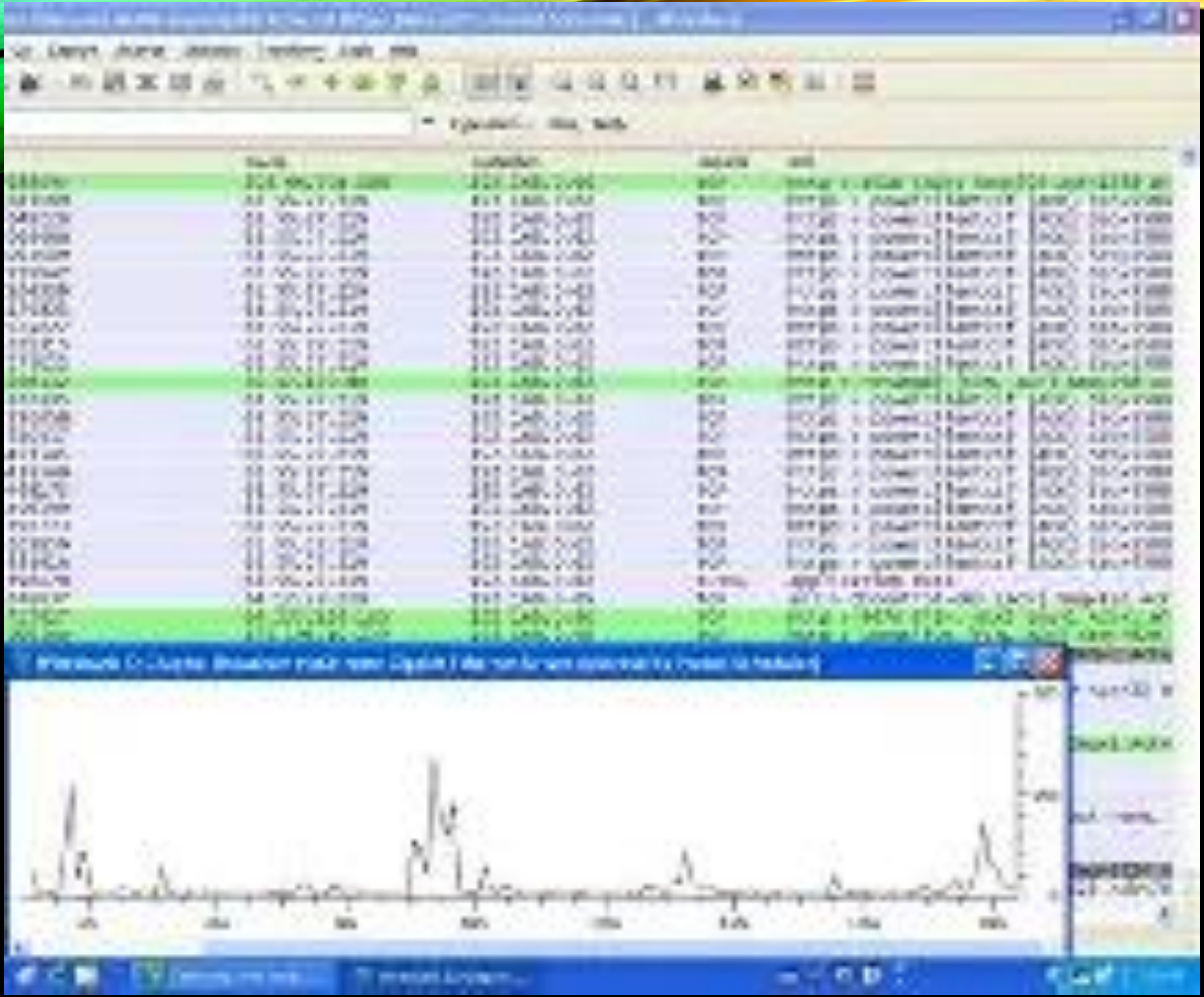
0
500
1000
1500
2000
2500
3000





```
             Source                  Destination        SrcPort   DstPort Pr
 32,278817   AsrockIn_52:f1:01       Broadcast                            AR
 32,278685   AsrockIn_52:f1:01       Broadcast                            AR
 32,400064   10.10.100.185           10.10.255.255        54915     54915 UD
 32,400046   10.10.100.185           10.10.255.255        54915     54915 UD
 33,053490   10.10.100.155           10.10.101.240        55703       445 TC
 33,278735   AsrockIn_52:f1:01       Broadcast                            AR
 33,278750   10.10.100.185           10.10.100.185        50000     50300 IC
 33,492343   10.10.100.185           10.10.255.255        54915     54915 UD
 33,492325   10.10.100.185           10.10.255.255        54915     54915 UD
 34,279151   10.10.100.155           10.10.100.155        51910      8002 IC
 34,494399   10.10.100.185           10.10.255.255        54915     54915 UD
 34,494413   10.10.100.185           10.10.255.255        54915     54915 UD
 35,492372   10.10.100.185           10.10.255.255        54915     54915 UD
 35,492349   10.10.100.185           10.10.255.255        54915     54915 UD
 36,373795   10.10.100.155           45.32.24.138         41954     55413 UD
 36,492869   10.10.100.185           10.10.255.255        54915     54915 UD
 36,492887   10.10.100.185           10.10.255.255        54915     54915 UD
 36,656549   45.32.24.130            10.10.100.185        55413     41954 UD
 37,534801   10.10.100.185           10.10.255.255        54915     54915 UD
```





THANK YOU!