National University of Computer and Emerging Sciences
Islamabad Campus

CLASS ACTIVITY

Developing a Sniffer

CS4061

Ethical Hacking Concepts & Practices

Submitted by: Muhammad Azan Afzal
Roll number: 22I-1741
Date: April 27, 2025

Table of Contents

• Introduction

Network traffic analysis and packet sniffing are crucial components of ethical hacking and network security. Packet sniffers allow security professionals to monitor and analyze network communication to detect vulnerabilities, diagnose network issues, and ensure system integrity. In this quiz, the objective was to develop a custom packet sniffing tool in Python capable of capturing, parsing, and optionally filtering network traffic from both wired (Ethernet) and wireless (Wi-Fi) interfaces. This practical exercise provided hands-on experience in packet analysis, the OSI model, and network troubleshooting methodologies within a secure and ethical environment.

• Steps

1. Objectives

- To design and implement a custom packet sniffing tool using Python.
- To capture all network traffic, including broadcast, unicast, and multicast packets, from both wired (Ethernet) and wireless (Wi-Fi) interfaces.
- To parse captured packets and analyze critical fields at different OSI layers.
- To log captured packet information for future analysis.
- To demonstrate optional filtering capabilities based on protocols like HTTP, DNS, ARP, TCP, etc.
- To ensure the tool operates ethically within a controlled test environment.

2. System Design

2.1 Technologies Used

- **Programming Language**: Python 3
- **Libraries**: Scapy (for packet capturing and parsing)
- **Operating System**: Kali Linux (root access utilized for sniffing)
- **Network Modes**:
  - **Monitor Mode** (for Wi-Fi interfaces)
  - **Promiscuous Mode** (for Ethernet interfaces)

2.2 Architecture Overview

- **User Interface**: Command-line based.
- **Interface Selection**: User provides the network interface (e.g., wlan0mon, eth0).
- **Packet Capturing**:
  - Sniff packets using Scapy's sniff() method.

- Capture all types of packets (broadcast, multicast, unicast).
- **Packet Parsing**:
  - Identify and extract fields from Ethernet, IP, TCP, UDP, and ARP layers.
  - Display source MAC/IP, destination MAC/IP, and protocol type.
- **Logging**:
  - Packet summary is written to captured_packets_log.txt file.
- **Filtering Support**:
  - Users can apply custom BPF filters to capture specific traffic types.
- **Graceful Exit**:
  - Handles KeyboardInterrupt (CTRL+C) and other runtime errors.

3. Code Overview

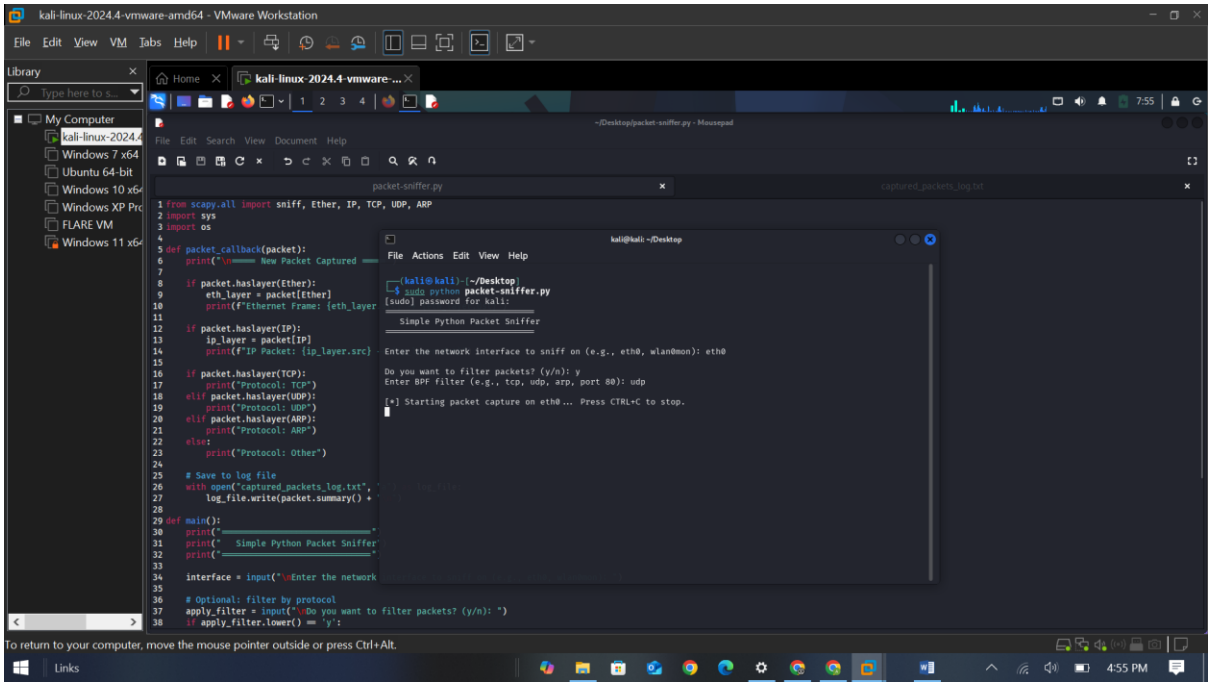
The code comprises the following main components:

- **Root Check**:
  - The tool checks if it is run with root privileges, essential for sniffing operations.
- **Main Function**:
  - Prompts the user for network interface.
  - Asks if the user wants to apply a BPF filter.
  - Initiates packet sniffing.
- **Packet Callback Function**:
  - For each captured packet:
    - Check and extract information from Ethernet, IP, TCP, UDP, and ARP layers.
    - Print relevant fields (e.g., source MAC/IP, destination MAC/IP, protocol type).
    - Append a summary of the packet to a text file.
- **Packet Sniffing**:
  - sniff(iface=interface, prn=packet_callback, filter=bpf_filter, store=0)
- **Error Handling**:
  - Displays meaningful errors if the interface is incorrect or if there are permission issues.

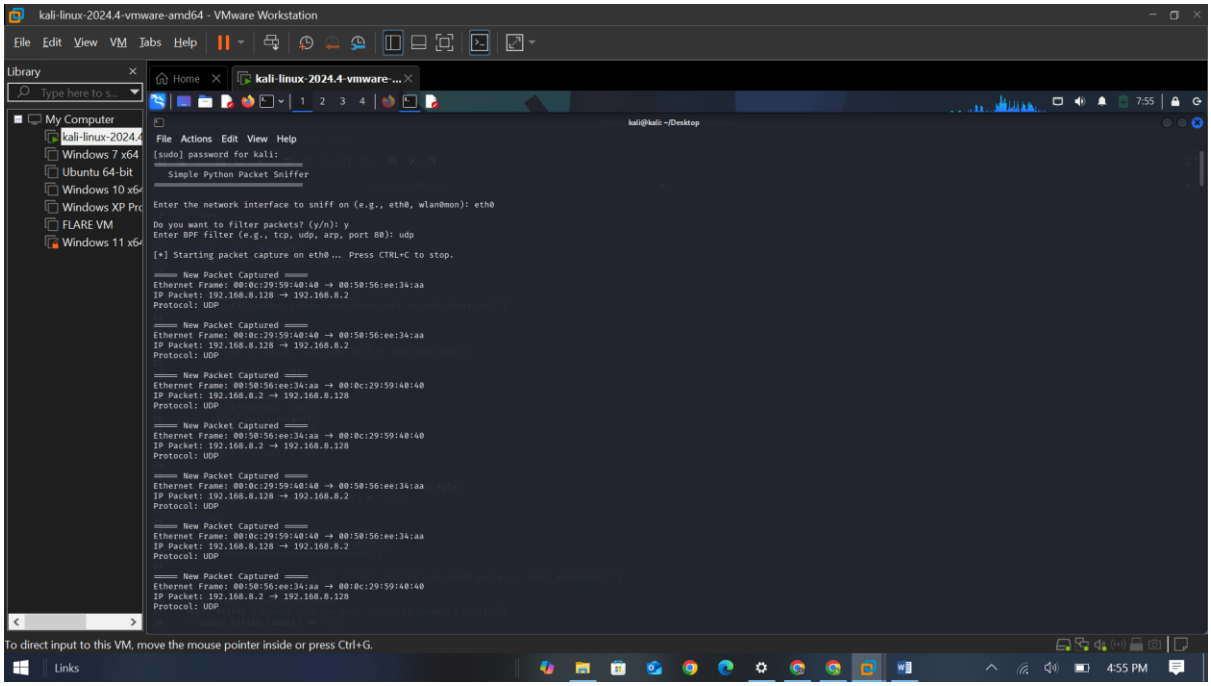
4. Screenshots and Testing Results

Testing was conducted in a controlled virtual lab setup using Kali Linux to generate network traffic.

4.1 Interface Selection Prompt

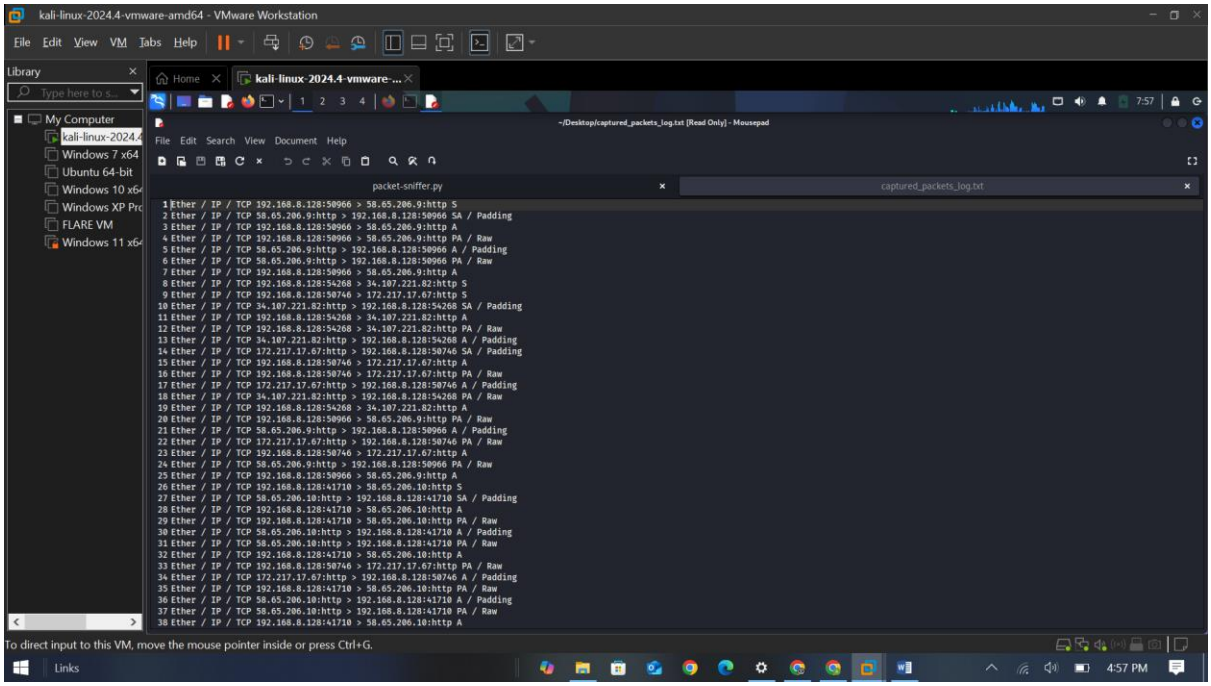


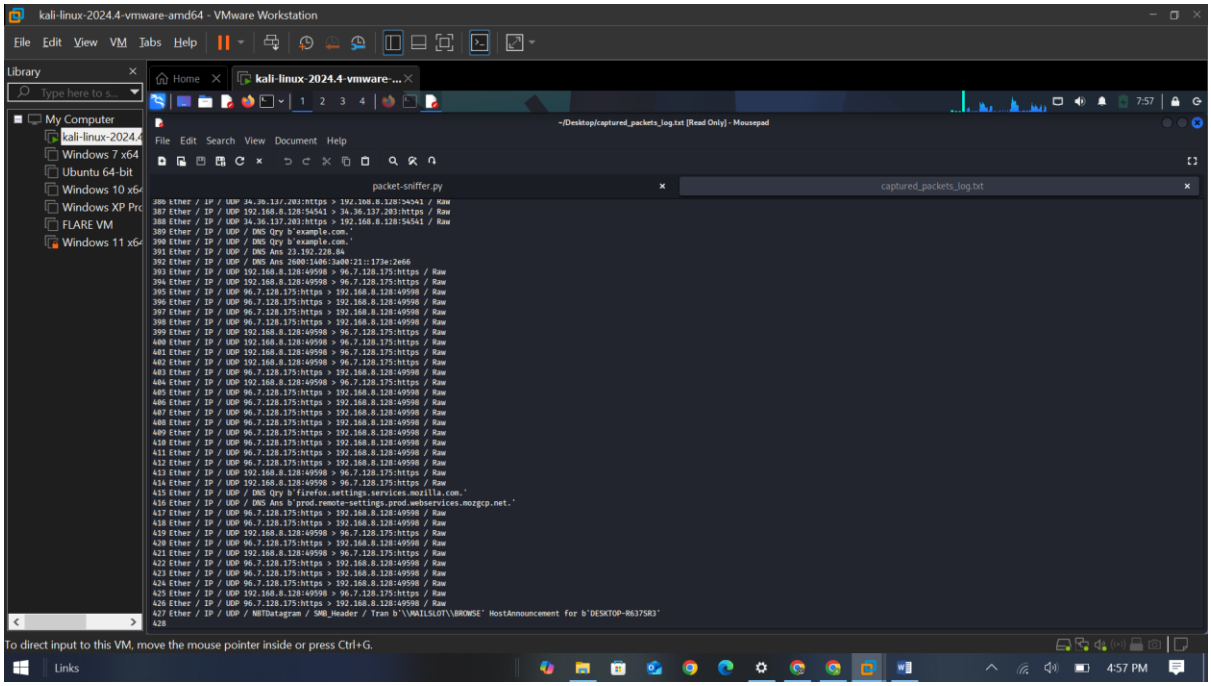
4.2 Packet Capture in Progress



Here I searched the website example.com and you can see that packets are captured by my tool.

4.3 Captured Packets Log File





5. Ethical Considerations

- The packet sniffer was exclusively used in an isolated, personal test lab setup.
- No traffic from unauthorized or external networks was captured.
- Monitor mode and promiscuous mode were enabled only on personally owned hardware.
- Capturing packets on public or unauthorized networks is illegal and unethical.
- The primary focus remained on educational and ethical hacking practices, in line with course guidelines.

6. Future Improvements

- **User Interface**:
  - Implement a graphical user interface (GUI) using Tkinter or PyQt5 for better usability.
- **Advanced Protocol Analysis**:
  - Reconstruct full HTTP sessions and extract login data (for ethical demonstration only).
- **Deep Packet Inspection**:
  - Analyze packet payloads in detail for specific application data.
- **Database Integration**:
  - Store captured packets and metadata into a SQL/NoSQL database for better searchability.
- **Real-Time Alerts**:
  - Notify user if suspicious traffic patterns are detected (e.g., ARP spoofing attempts).
- **Wireless Management Frame Detection**:
  - Detect Wi-Fi probe requests, beacon frames, and disassociation frames for wireless reconnaissance.

• CONCLUSION

The packet sniffer tool was successfully developed and demonstrated. It captured real-time traffic from both wired and wireless interfaces, parsed key layers, displayed critical information, and logged the packet summaries. The project helped reinforce key networking concepts like OSI layers, protocols, and packet structures. All operations were conducted ethically within isolated testing environments. Future improvements can make the tool more robust, user-friendly, and powerful for advanced network security analysis.

References

- Scapy Documentation. (n.d.). Retrieved from: https://scapy.readthedocs.io/
- Kurose, J. F., & Ross, K. W. (2017). Computer Networking: A Top-Down Approach (7th Edition). Pearson.

End of Report