CS-221

Project report

Network Packet Sniffer Using Python

Team Name: Packet watch

Members:

-Faizan Ali (2023192)

-Aayan Rashid (2023002)

Section: E (CYS)

Abstract:

PacketWatch is an advanced network packet capture and analysis tool designed to provide real-time insights into network traffic. Leveraging a robust graphical user interface (GUI) built with the Tkinter library, PacketWatch employs various data structures such as linked lists, stacks, AVL trees, priority queues, tries, graphs, and deques to efficiently manage and analyze captured packet data. Utilizing the Scapy library for packet capturing, this tool aims to offer network administrators and security professionals a comprehensive solution for monitoring, analyzing, and securing network communications

Introduction:

The purpose of this project is to develop a comprehensive packet sniffing tool that can capture and analyze network traffic in real-time. PacketWatch is designed to help network administrators and security professionals monitor network activity, identify potential issues, and analyze traffic patterns. By leveraging Python and the Tkinter library, the project provides an intuitive interface for users to interact with the captured data. The importance of this tool lies in its ability to provide real-time insights and enhance network security through effective monitoring and analysis.

System Requirements:

Hardware:

- A computer with a network interface card (NIC) capable of promiscuous mode.
- Minimum 4GB RAM and 2GHz processor.

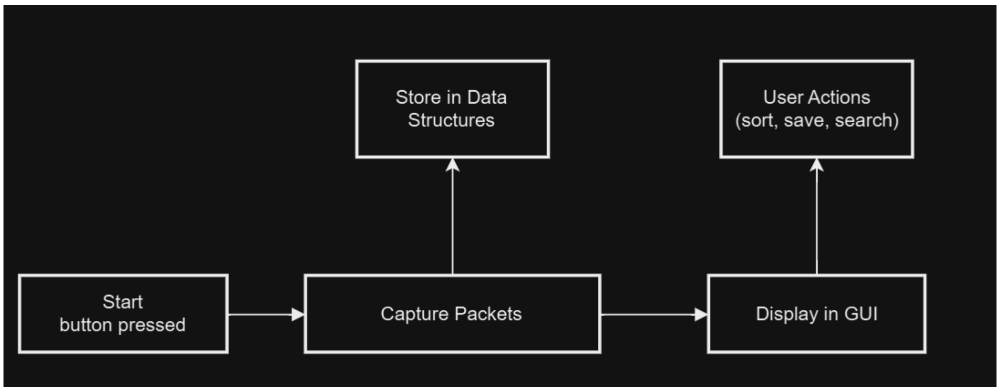
Software:

- Operating System: Windows, macOS, or Linux.
- Python 3.x.
- Required Python Libraries: tkinter, scapy, threading, heapq.

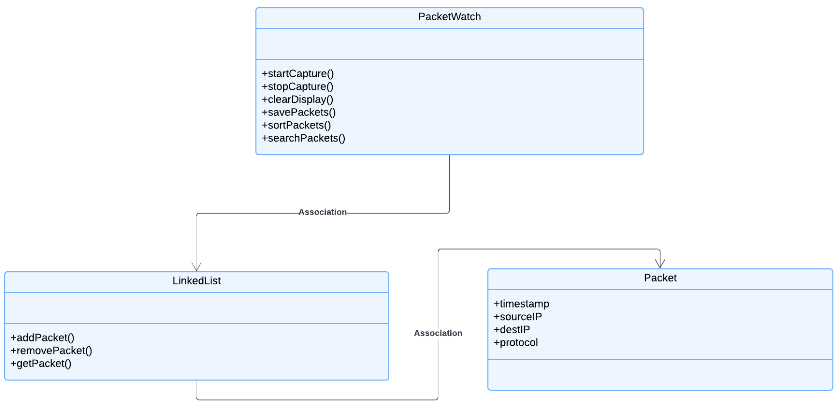
Dependencies:

- Scapy library for packet capture and analysis.
- Tkinter for the graphical user interface.

System Design flowchart:



UML Class Diagram:



Objectives:

1. **Develop a User-Friendly GUI**:
   - Provide an intuitive interface for users to start, stop, clear, save, sort, and search captured packets.
2. **Implement Robust Data Structures**:
   - Utilize various data structures to ensure efficient storage, retrieval, and manipulation of packet data.
3. **Real-Time Packet Capture**:
   - Enable real-time packet capture and display the captured packets in the GUI.
4. **Advanced Packet Management**:
   - Implement features such as sorting, searching, and saving packets to enhance usability.

Implementation Details:

1. GUI Development:

- **Tkinter Library**: Used to create a user-friendly interface with buttons and a scrolled text area for easy interaction.
- **Buttons**: Include functionalities for starting, stopping, clearing, saving, sorting, and searching packets to ensure comprehensive user control over packet capture and analysis.
- **Scrolled Text Area**: Displays captured packets with support for text highlighting, allowing users to easily view and analyze packet information.

2. Data Structures:

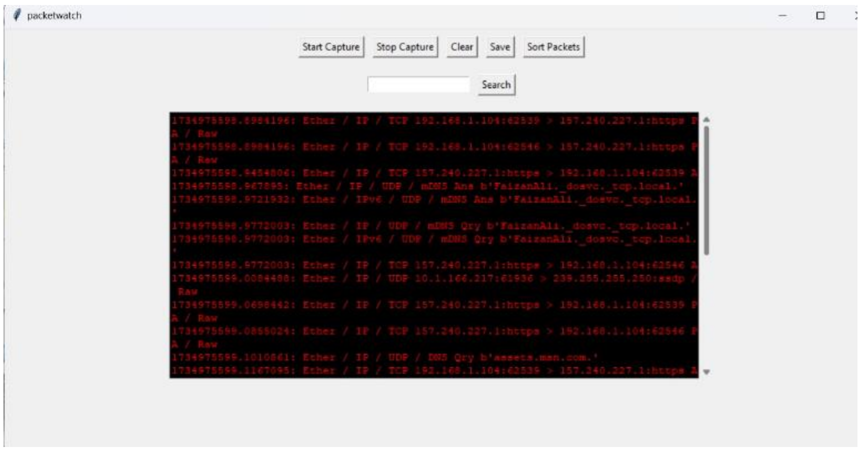
- **Linked List**: Used for sequential storage of packets, ensuring quick insertion and traversal.
- **Stack**: Stores recent packets with Last-In-First-Out (LIFO) access, useful for quickly accessing the latest packets.
- **Priority Queue**: Manages packets based on priority (timestamp), ensuring efficient retrieval of the most relevant packets.
- **Trie**: Implements efficient search of packet prefixes (e.g., IP addresses), enabling quick lookup of specific packets.

- **Graph**: Represents device communication, allowing visualization and analysis of network relationships.
- **Deque**: Double-ended queue for managing recent packets with both First-In-First-Out (FIFO) and LIFO access.
- **AVL Tree**: Self-balancing tree for balanced storage and quick search of packets by timestamp.

3. Packet Capture:

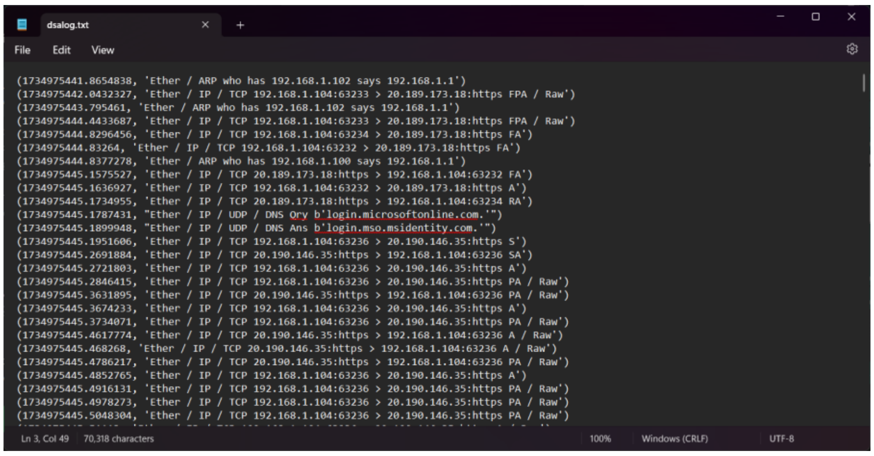
- **Scapy Library**: Utilized for capturing network packets and processing packet data. Scapy's flexibility allows for detailed packet manipulation and analysis.
- **Multithreading**: Ensures the GUI remains responsive during packet capture by running the capture process in a separate thread. This prevents the application from freezing and provides a smooth user experience.

Main GUI interface:



```
1734975598.8984196: Ether / IP / TCP 192.168.1.104:62539 > 157.240.227.1:https P
A / Raw
1734975598.8984196: Ether / IP / TCP 192.168.1.104:62546 > 157.240.227.1:https P
A / Raw
1734975598.9454806: Ether / IP / TCP 157.240.227.1:https > 192.168.1.104:62539 A
1734975598.967895: Ether / IP / UDP / mDNS Ans b'FaizanAli._dosvc._tcp.local.'
1734975598.9721932: Ether / IPv6 / UDP / mDNS Ans b'FaizanAli._dosvc._tcp.local.
'
1734975598.9772003: Ether / IP / UDP / mDNS Qry b'FaizanAli._dosvc._tcp.local.'
1734975598.9772003: Ether / IPv6 / UDP / mDNS Qry b'FaizanAli._dosvc._tcp.local.
'
1734975598.9772003: Ether / IP / TCP 157.240.227.1:https > 192.168.1.104:62546 A
1734975599.0084488: Ether / IP / UDP 10.1.166.217:61936 > 239.255.255.250:ssdp /
 Raw
1734975599.0698442: Ether / IP / TCP 157.240.227.1:https > 192.168.1.104:62539 P
A / Raw
1734975599.0855024: Ether / IP / TCP 157.240.227.1:https > 192.168.1.104:62546 P
A / Raw
1734975599.1010861: Ether / IP / UDP / DNS Qry b'assets.msn.com.'
1734975599.1167095: Ether / IP / TCP 192.168.1.104:62539 > 157.240.227.1:https A
```

Saved text file:



```
(1734975441.8654838, 'Ether / ARP who has 192.168.1.102 says 192.168.1.1')
(1734975442.0432327, 'Ether / IP / TCP 192.168.1.104:63233 > 20.189.173.18:https FPA / Raw')
(1734975443.795461, 'Ether / ARP who has 192.168.1.102 says 192.168.1.1')
(1734975444.4433687, 'Ether / IP / TCP 192.168.1.104:63233 > 20.189.173.18:https FPA / Raw')
(1734975444.8296456, 'Ether / IP / TCP 192.168.1.104:63234 > 20.189.173.18:https FA')
(1734975444.83264, 'Ether / IP / TCP 192.168.1.104:63232 > 20.189.173.18:https FA')
(1734975444.8377278, 'Ether / ARP who has 192.168.1.100 says 192.168.1.1')
(1734975445.1575527, 'Ether / IP / TCP 20.189.173.18:https > 192.168.1.104:63232 FA')
(1734975445.1636927, 'Ether / IP / TCP 192.168.1.104:63232 > 20.189.173.18:https A')
(1734975445.1734955, 'Ether / IP / TCP 20.189.173.18:https > 192.168.1.104:63234 RA')
(1734975445.1787431, "Ether / IP / UDP / DNS Qry b'login.microsoftonline.com.'")
(1734975445.1899948, "Ether / IP / UDP / DNS Ans b'login.mso.msidentity.com.'")
(1734975445.1951606, 'Ether / IP / TCP 192.168.1.104:63236 > 20.190.146.35:https S')
(1734975445.2691884, 'Ether / IP / TCP 20.190.146.35:https > 192.168.1.104:63236 SA')
(1734975445.2721803, 'Ether / IP / TCP 192.168.1.104:63236 > 20.190.146.35:https A')
(1734975445.2846415, 'Ether / IP / TCP 192.168.1.104:63236 > 20.190.146.35:https PA / Raw')
(1734975445.3631895, 'Ether / IP / TCP 20.190.146.35:https > 192.168.1.104:63236 PA / Raw')
(1734975445.3674233, 'Ether / IP / TCP 192.168.1.104:63236 > 20.190.146.35:https A')
(1734975445.3734071, 'Ether / IP / TCP 192.168.1.104:63236 > 20.190.146.35:https PA / Raw')
(1734975445.4617774, 'Ether / IP / TCP 20.190.146.35:https > 192.168.1.104:63236 A / Raw')
(1734975445.468268, 'Ether / IP / TCP 20.190.146.35:https > 192.168.1.104:63236 A / Raw')
(1734975445.4786217, 'Ether / IP / TCP 20.190.146.35:https > 192.168.1.104:63236 PA / Raw')
(1734975445.4852765, 'Ether / IP / TCP 192.168.1.104:63236 > 20.190.146.35:https A')
(1734975445.4916131, 'Ether / IP / TCP 192.168.1.104:63236 > 20.190.146.35:https PA / Raw')
(1734975445.4978273, 'Ether / IP / TCP 192.168.1.104:63236 > 20.190.146.35:https PA / Raw')
(1734975445.5048304, 'Ether / IP / TCP 192.168.1.104:63236 > 20.190.146.35:https PA / Raw')
```

Key Features:

1. **Start/Stop Capture**:
   - Users can start and stop packet capture with the click of a button, providing control over the capture process.
2. **Clear Text Area**:
   - Clear captured packet data from the display, allowing users to reset the analysis environment quickly.
3. **Save Captured Packets**:
   - Save captured packet data to a text file for later analysis, ensuring data persistence and accessibility.
4. **Sort Packets**:
   - Sort captured packets based on timestamp, making it easier to analyze the packet flow over time.
5. **Search Functionality**:
   - Search for specific packets using a trie data structure, enabling efficient and fast lookup of packet information.

User Guide:

Installation:

1. Install Python 3.x from the official website.
2. Install required libraries using pip: “pip install scapy”

Configuration:

- No special configuration required. Ensure the network interface card is in promiscuous mode if needed.

Running the Project:

1. Run the Python script to start the PacketWatch application: “python packetwatch.py”
2. Use the GUI buttons to start/stop packet capture, clear the display, save captured packets, sort packets, and search for specific packets.

Observations and Challenges:

- **Performance Optimization**: Managing a large volume of packets required careful consideration of data structures to ensure performance was not degraded. Optimizations were made to handle high packet rates efficiently.
- **Error Handling**: Ensuring robust error handling for file operations and network capture was critical to prevent crashes and data loss. Comprehensive error messages and recovery mechanisms were implemented.
- **User Feedback**: Providing real-time feedback to users during lengthy operations, such as saving or sorting packets, improved the user experience. Progress indicators and status messages were added.

Future Enhancements:

1. **Real-Time Packet Analysis**:
   - Implement real-time analysis to detect potential security threats or anomalies, providing immediate insights and alerts.
2. **Visualization**:
   - Add graphical visualizations of network traffic patterns and device communication, enabling users to see network dynamics at a glance.
3. **Advanced Filtering**:
   - Provide advanced filtering options for packet capture based on various criteria (e.g., protocol type, source/destination IP), enhancing the tool's flexibility.
4. **Reporting**:
   - Generate detailed reports with summaries and statistics of captured packets, offering users valuable insights into network behavior.
5. **Integration with Other Tools**:
   - Integrate with established network analysis tools (e.g., Wireshark) for extended functionality, allowing users to leverage additional capabilities.
6. **Security Enhancements**:
   - Implement user authentication and access control to restrict access to sensitive data and application features, ensuring only authorized users can use the tool.

Conclusion:

PacketWatch successfully provides a comprehensive solution for network packet capture and analysis. The combination of a user-friendly GUI, robust data structures, and real-time packet processing ensures that users can efficiently manage and analyze network traffic. Future enhancements will further extend the tool’s capabilities, making it an indispensable resource for network administrators and security professionals.

References:

- **Scapy Documentation**:
  - Scapy Documentation. Available at: https://scapy.readthedocs.io/.

-**Python Tkinter Documentation**:
  - Python Software Foundation. Tkinter — Python Interface to Tcl/Tk. Available at: https://docs.python.org/3/library/tkinter.html.

-**Multithreading in Python**:
  - Python Software Foundation. threading — Thread-based Parallelism. Available at: https://docs.python.org/3/library/threading.html.

-**Data Structures and Algorithms in Python**:
  - Real Python. Data Structures and Algorithms in Python. Available at: https://realpython.com/data-structures-algorithms-python/.