Network Packet Analyzer using Python

Author: Deepak M

Institution / Internship Company: Prodigy

Abstract

This project focuses on building a lightweight Network Packet Analyzer using Python. The tool leverages raw sockets and the DPKT library to capture and decode Ethernet, IP, TCP, and UDP packets in real-time. The goal is to understand how low-level packet sniffing works in Linux and demonstrate packet parsing for cybersecurity and networking educational purposes.

Introduction

Packet sniffing is the process of monitoring and capturing network packets passing through a network interface. Network analyzers are essential tools used in penetration testing, cybersecurity research, traffic debugging, and network diagnostics. This project implements a basic packet analyzer that works in Linux using AF_PACKET raw sockets and the DPKT parsing library.

System Requirements

- Python 3.x

- Linux operating system (Kali recommended)

- Root privileges

- Python libraries: dpkt

Architecture

The architecture consists of three major layers:
1. Capture Layer – Uses raw sockets to receive raw Ethernet frames.
2. Parsing Layer – Uses DPKT to decode Ethernet, IP, TCP, and UDP headers.
3. Output Layer – Displays parsed information to the terminal in real-time.

Methodology

The tool opens a raw socket using AF_PACKET in promiscuous mode, allowing the program to capture every incoming and outgoing packet. The raw binary data is then passed to DPKT, which converts it into Python-readable objects representing Ethernet, IP, TCP, and UDP structures. The tool extracts fields such as MAC addresses, IP addresses, ports, flags, and sequence numbers.

Implementation

The packet sniffer was implemented in Python using dpkt for protocol parsing. Raw sockets were used to capture packets, and each packet was decoded and printed with timestamp, MAC addresses, IP addresses, and protocol-level details.

Sample Code Snippet:

```
sock = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(3))
raw_data, addr = sock.recvfrom(65535)
eth = dpkt.ethernet.Ethernet(raw_data)
```

Testing & Results

The tool successfully captured and parsed packets in real-time. TCP and UDP traffic was decoded accurately, showing source/destination MAC addresses, IP addresses, ports, sequence numbers, and flags. The tool handled malformed packets gracefully using try/except blocks.

Future Scope

- Save packets in PCAP format
- Build GUI version (Tkinter/PyQT)
- Add support for DNS, HTTP, TLS parsing
- Add filtering (similar to Wireshark display filters)
- Implement multithreaded performance optimization

Conclusion

This project successfully demonstrates how Python can be used to build a basic network packet analyzer using raw sockets and protocol parsing libraries. It serves as a strong foundational project for internships, cybersecurity portfolios, and further network tool development.

References

- Python dpkt documentation

- Linux raw socket documentation

- Network protocol RFCs