Real-Time Network Traffic Sniffer with Anomaly Detection

Abstract

This project involves the development of a real-time packet sniffer with anomaly detection using Python. The tool captures live traffic, stores relevant metadata into an SQLite database, and monitors for network anomalies such as flooding and port scanning. It provides a simple Command-Line Interface (CLI) for configuration, logs alerts into a file, and allows post-analysis via SQL queries.

Tools Used

- Python 3.x
- Scapy (for packet sniffing)
- SQLite (for logging and analysis)

Steps Involved in Building the Project

1. Installed required libraries: scapy and sqlite3 (built-in).
2. Initialized an SQLite database with tables for storing packet metadata and anomaly alerts.
3. Captured live packets using scapy and logged details (timestamp, source IP, destination IP, ports, length, flags).
4. Implemented anomaly detection logic for flooding (packet/byte thresholds) and port scanning.
5. Logged alerts in both SQLite and a separate alerts.log file.
6. Enabled configurable thresholds and interface selection through CLI arguments.
7. Provided queries for post-analysis and traffic summaries from the database.

Usage Options

Run the tool with administrator privileges.

Example:
python3 packet_sniffer.py --iface "Ethernet" --window 10 --pkt-th 2000 --byte-th 5000000 --portscan 100

Options:
--iface : Network interface name (e.g., Ethernet, Wi-Fi, eth0)
--window : Sliding window in seconds (default: 10)

--pkt-th : Packet flood threshold per source (default: 2000)
--byte-th : Byte flood threshold per source (default: 5MB)
--portscan : Distinct ports threshold for port scanning (default: 100)
--db : Path to SQLite DB file (default: traffic.db)
--alerts : Path to alerts log file (default: alerts.log)

Database and Logs Usage

Database (traffic.db):
- Packets table: Stores packet metadata for analysis.
- Alerts table: Stores anomaly alerts with type, source, and details.

Example SQL queries:
- SELECT src_ip, COUNT(*) FROM packets GROUP BY src_ip ORDER BY COUNT(*) DESC;
- SELECT type, COUNT(*) FROM alerts GROUP BY type;

Logs (alerts.log):
- Contains real-time anomaly alerts appended during execution.
- Can be monitored live using: tail -f alerts.log (Linux) or Get-Content alerts.log -Wait (Windows).

Conclusion

The CLI-based packet sniffer with anomaly detection provides a practical solution for monitoring network traffic in real-time. It combines live capture, database logging, anomaly detection, and reporting into a single tool. The project demonstrates the use of Python for applied cybersecurity tasks and offers a foundation for more advanced IDS/IPS solutions.