# Advanced Network Packet Sniffer with

# AI-Based Threat Detection

## Table of Contents

|  |  |
| --- | --- |
|  | Introduction |
|  | Objectives |
|  | System Requirements  3.1 Hardware Requirements  3.2 Software Requirements |
|  | Features and Functionality |
|  | System Architecture |
|  | Step-by-Step Implementation Guide |
|  | Real-World Applications |
|  | Testing and Evaluation |
|  | Future Enhancements |
|  | Conclusion |

## 1. Introduction

This project is an Advanced Network Packet Sniffer designed to capture, analyze, and visualize network traffic. It includes AI-based threat detection, IDS alerts, and packet injection techniques for security testing. The application features a futuristic GUI with real-time data analysis and filtering.

## ****2. Objectives****

The main objectives of this project are:

* **Capture and analyze real-time network packets.**
* **Implement AI-based threat detection to identify potential cyber threats.**
* **Provide IDS alerts for malicious activity.**
* **Allow packet filtering and traffic visualization with interactive graphs.**
* **Enable packet injection (SYN Flood, ICMP Ping) for penetration testing.**
* **Design a modern, user-friendly GUI for better usability.**

## ****3. System Requirements****

### ****3.1 Hardware Requirements****

* **Processor:** Intel Core i5 or higher
* **RAM:** 8GB minimum (16GB recommended)
* **Storage:** At least 500MB of free space
* **Network Adapter:** Supports packet capture (WinPcap/Npcap for Windows)

### ****3.2 Software Requirements****

* **Operating System:** Windows 10/11, Linux, or macOS
* **Python Version:** Python 3.9 or later (tested on Python 3.13)
* **Required Libraries:**

## ****4. Features and Functionality****

* **Packet Sniffing with Scapy**
* **AI-Based Threat Detection**
* **IDS Alerts for Intrusion Detection**
* **Packet Filtering for Protocols**
* **Real-Time Traffic Visualization**
* **Packet Injection (SYN Flood, ICMP Ping)**
* **Auto-Detect Network Interfaces**
* **Modern GUI with Dark Theme**

## ****5. System Architecture****

The **Advanced Network Packet Sniffer with AI-Based Threat Detection** follows a modular architecture that ensures efficient **packet capturing, analysis, visualization, and security assessment**. The system is structured into three primary layers:

1. **User Interface (GUI) Layer**
2. **Backend Processing Layer**
3. **Network Communication Layer**

### ****5.1 User Interface (GUI) Layer****

This layer provides an **intuitive and user-friendly interface** for interacting with the packet sniffer. It is developed using **Tkinter** and allows users to:

* Start and stop packet sniffing.
* Select a **network interface** from the available options.
* Apply **filters** to view specific types of packets (TCP, UDP, ICMP, etc.).
* View **real-time network traffic graphs**.
* Inject packets for **security testing**.
* Receive **alerts from the Intrusion Detection System (IDS)**.

#### ****GUI Functional Components:****

* **Main Dashboard:** Displays live network traffic and statistics.
* **Filter Options:** Allows filtering packets by protocol or source/destination.
* **Threat Alerts:** Shows security warnings when suspicious activity is detected.
* **Graphical Visualizations:** Provides insights into network traffic trends.
* **Packet Injection Panel:** Enables users to perform SYN Flood or ICMP Ping attacks.

### ****5.2 Backend Processing Layer****

This layer manages **packet capturing, processing, and AI-based threat analysis**. It consists of several key components:

#### ****Packet Sniffing (Using Scapy)****

* Captures real-time network traffic using **Scapy**.
* Extracts essential details such as **source/destination IP, protocol type, packet size, etc.**
* Temporarily stores packets for analysis and visualization.

📌 **Example Code for Packet Sniffing:**

python

CopyEdit

scapy.sniff(iface=interface, prn=self.packet\_callback, store=False)

#### ****AI-Based Threat Detection****

* Uses **machine learning models** to analyze packet behavior.
* Extracts features such as **packet length, time intervals, and protocol usage**.
* The **trained AI model** predicts whether the packet is **malicious**.
* If a **threat is detected**, an alert is generated.

📌 **Example Code for Threat Detection:**

python

CopyEdit

features = np.array([[len(packet), packet.time % 1]])

prediction = ai\_model.predict(features)

if prediction[0] == 1:

messagebox.showwarning('Threat Detected', 'Suspicious activity detected!')

#### ****Intrusion Detection System (IDS)****

* Monitors network traffic for **known attack patterns**.
* If a packet matches a **suspicious signature**, an **alert is displayed**.

📌 **Example IDS Alert Code:**

python

CopyEdit

if 'malicious' in packet.summary().lower():

messagebox.showwarning('Intrusion Detected', 'Potential Threat Detected!')

#### ****Traffic Visualization****

* Uses **Matplotlib and Seaborn** to generate **real-time graphs**.
* Displays **traffic statistics** based on **protocol distribution**.

📌 **Example Code for Graph Generation:**

python

CopyEdit

sns.barplot(x=list(protocol\_counts.keys()), y=list(protocol\_counts.values()), palette='coolwarm')

plt.show()

#### ****Packet Injection (Penetration Testing)****

* Supports **SYN Flood** and **ICMP Ping injection** for **penetration testing**.

📌 **Example Code for SYN Flood Attack:**

python

CopyEdit

ip\_layer = scapy.IP(src='192.168.1.' + str(random.randint(2, 254)), dst=target\_ip)

tcp\_layer = scapy.TCP(sport=random.randint(1024, 65535), dport=target\_port, flags='S')

packet = ip\_layer / tcp\_layer

scapy.send(packet, verbose=False)

### ****📌 System Architecture Diagram:****

pgsql

CopyEdit

+--------------------------------------------------+

| Advanced Packet Sniffer GUI |

+--------------------------------------------------+

| User Interface (Tkinter) |

| - Start/Stop Sniffing |

| - Select Network Interface |

| - Apply Filters |

| - View Traffic Graphs |

| - Perform Packet Injection |

+--------------------------------------------------+

| Backend Components |

| - Packet Sniffing (Scapy) |

| - Threat Detection (AI Model) |

| - Intrusion Detection System (IDS) |

| - Data Visualization (Matplotlib/Seaborn) |

+--------------------------------------------------+

| Network Layer (TCP/IP Packets) |

| - Capturing Incoming & Outgoing Packets |

| - Filtering and Analyzing Traffic |

| - Detecting Threats |

+--------------------------------------------------+

## 6. Step-by-Step Implementation Guide

Step 1: Install Required Dependencies  
Open a terminal or command prompt and run the following command:

pip install scapy matplotlib psutil pyshark pandas numpy joblib seaborn tk

Step 2: Run the Packet Sniffer  
Execute the script by running:

python sniffer\_gui.py

Step 3: Select a Network Interface  
The application will automatically detect available network interfaces. Select one from the list to start sniffing.

Step 4: Capture and Analyze Packets  
Once started, the GUI will display live network traffic. Users can filter packets by protocol (TCP, UDP, ICMP) and view their details.

Step 5: Perform AI-Based Threat Detection  
The system analyzes network patterns using an AI model and alerts users about suspicious activity.

Step 6: Inject Packets for Security Testing  
The tool supports SYN Flood and ICMP Ping attacks to test network resilience. Users can trigger these attacks via the GUI.

Step 7: View Graphical Analysis  
Real-time graphs display network traffic statistics, making it easier to analyze trends and detect anomalies.

## 7. Testing and Evaluation

|  |  |  |
| --- | --- | --- |
| **Feature** | **Expected Outcome** | **Test Status** |
| **Start Packet Sniffing** | **Packets captured and displayed** | **✅ Pass** |
| **Stop Packet Sniffing** | **Packet capture stops** | **✅ Pass** |
| **Apply Filter (TCP)** | **Only TCP packets Displayed** | **✅ Pass** |
| **AI-Based Threat Detection** | **IDS alerts if suspicious activity detected** | **✅ Pass** |
| **Show Traffic Graph** | **Graph displays network activity** | **✅ Pass** |
| **Perform SYN Flood** | **Sends multiple SYN packets** | **✅ Pass** |

## ****8. Real-World Applications****

• **Network Security Analysis:** Detect and prevent cyber threats in real time.  
• **Ethical Hacking:** Penetration testers can use packet injection to simulate attacks.  
• **Traffic Monitoring:** IT professionals can analyze network traffic trends for optimization.  
• **Education & Research:** Used in cybersecurity courses to demonstrate packet analysis techniques.  
• **Intrusion Detection:** Helps security teams identify unauthorized access attempts.

## 9. Future Enhancements

• Integration with Web Dashboard for remote monitoring.  
• AI Model Improvement using larger datasets.  
• Automated Threat Response (e.g., blocking malicious IPs).  
• Multi-Protocol Support for deeper analysis.  
• Support for cloud-based threat intelligence.

## 10. Conclusion

The Advanced Network Packet Sniffer is a powerful security tool that helps capture, analyze, and detect network threats in real-time. With AI-based threat detection, packet injection, and IDS alerts, it provides a robust solution for cybersecurity professionals, network administrators, and researchers. Future enhancements will further expand its capabilities, making it a versatile tool for network security.

## 11. ****References****

1. Scapy Documentation - Packet Manipulation Library  
   [*https://scapy.readthedocs.io/*](https://scapy.readthedocs.io/)
2. Python Official Documentation - Network Programming  
   [*https://docs.python.org/3/library/socket.html*](https://docs.python.org/3/library/socket.html)
3. Nmap Network Scanner - Security Auditing Tool  
   [*https://nmap.org/*](https://nmap.org/)
4. Wireshark User Guide - Network Traffic Analysis  
   https://www.wireshark.org/docs/
5. Intrusion Detection Systems (IDS) - SANS Institute  
   https://www.sans.org/white-papers/ids/
6. Matplotlib and Seaborn for Data Visualization  
   [*https://matplotlib.org/*](https://matplotlib.org/)  
   https://seaborn.pydata.org/
7. Machine Learning for Cybersecurity - Research Paper  
   [*https://arxiv.org/abs/1901.08867*](https://arxiv.org/abs/1901.08867)
8. Penetration Testing with Python - Offensive Security  
   [*https://www.offensive-security.com/*](https://www.offensive-security.com/)
9. AI-Based Threat Detection in Network Security  
   [*https://ieeexplore.ieee.org/document/8686026*](https://ieeexplore.ieee.org/document/8686026)
10. Network Security Best Practices - OWASP  
    <https://owasp.org/www-project-top-ten/>

## 12.Credits

Rajavarman.G.R

Linkedin:- <https://www.linkedin.com/in/rajavarman-g-r/>

Github:- <https://github.com/Rajavarman-GR>