**📄 AI-Based Real-Time Intrusion Detection System (IDS)**

**🔍 Project Overview**

This project implements a **simple, AI-powered Intrusion Detection System (IDS)** that monitors network traffic in real-time, extracts key features from packets, and uses a pre-trained **machine learning model** to classify traffic as **normal or malicious**.

The system uses the **NSL-KDD** dataset for training and operates in **real time using Scapy** for packet capture.

**🎯 Objectives**

* Capture network packets in real time.
* Extract relevant features that represent packet behavior.
* Train a machine learning model using historical network data.
* Detect potential intrusions and log alerts in real time.
* Keep the system lightweight and easy to deploy.

**🛠️ Tools & Technologies Used**

| **Technology** | **Purpose** |
| --- | --- |
| **Python** | Programming language |
| **Scapy** | Real-time packet sniffing |
| **pandas** | Data processing |
| **scikit-learn** | Machine learning model training |
| **joblib** | Model serialization |
| **NSL-KDD** | Benchmark intrusion detection dataset |

**📁 Dataset: NSL-KDD**

We used the KDDTrain+.txt and KDDTest+.txt files from the NSL-KDD dataset, which contain labeled network traffic records with 41 features and a class label (normal or attack type).

**✅ Preprocessing Steps**

1. Dropped duplicates and nulls.
2. Encoded categorical columns: protocol\_type, service, flag.
3. Simplified to **5 core features** for real-time use:
   * protocol\_type
   * service
   * flag
   * src\_bytes
   * dst\_bytes
4. Combined all attack types into a single label: "attack".

**🧠 Model Training (train\_model.py)**

We trained a **Random Forest Classifier** using the simplified features above.

**Steps:**

1. Read dataset.
2. Extract selected features and target labels.
3. Encode categorical features using LabelEncoder.
4. Train RandomForestClassifier.
5. Save model and encoders as ids\_model\_simple.pkl using joblib.

**🌐 Real-Time Packet Monitoring (capture.py + detect\_intrusions.py)**

**capture.py**

* Uses Scapy to sniff packets with IP + TCP/UDP/ICMP layers.
* Calls the detect() function for each relevant packet.

**detect\_intrusions.py**

* Extracts 5 features from each packet.
* Applies the same encoding as during training.
* Loads trained model and predicts label.
* If attack, logs an alert with emoji and timestamp.

**Example Alert:**

[2025-09-14 20:18:52.735518] 🚨 INTRUSION DETECTED! Features: [0, 44, 9, 0, 0]

Alerts are saved to:

logs/intrusion\_logs.txt

**🧪 Testing & Validation**

**✅ Simulated Intrusions**

We tested detection using:

* **nmap SYN scans** (nmap -sS 127.0.0.1)
* **UDP flood** using custom Python script
* **Custom crafted packets** (e.g., via hping3)

The system successfully detected and logged suspicious behavior based on learned patterns from the NSL-KDD dataset.

**📂 Project Structure**

project/

├── data/

│ └── NSL-KDD/

│ └── KDDTrain+.txt, KDDTest+.txt, KDDFeatureNames.txt

├── logs/

│ └── intrusion\_logs.txt

├── model/

│ └── ids\_model\_simple.pkl

├── src/

│ ├── train\_model.py

│ ├── capture.py

│ └── detect\_intrusions.py

├── requirements.txt

└── readme.md

**▶️ How to Run**

**1. Install Dependencies**

pip install -r requirements.txt

Make sure you have scapy, pandas, scikit-learn, joblib installed.

**2. Train the Model (only once)**

python src/train\_model.py

**3. Start Real-Time IDS**

python src/capture.py

**4. Generate Traffic to Test**

# Example: run this in another terminal or machine

nmap -sS 127.0.0.1

Check logs in:

logs/intrusion\_logs.txt

**⚠️ Known Warnings & Fixes**

| **Warning/Error** | **Fix** |
| --- | --- |
| X has 5 features, but model expects 41 | Simplified model uses only 5 features |
| charmap codec can't encode character | Resolved by writing logs with encoding='utf-8' |
| No libpcap provider available | Safe to ignore on Windows; packet capture still works |

**🚀 Future Improvements**

* Use full 41-feature model for improved accuracy.
* Integrate with **firewall** to auto-block IPs.
* Add **web dashboard** using Flask or Streamlit.
* Add **Slack/email alert integration**.
* Use **flow-based features** (not just per-packet).
* Run as a Windows/Linux **background service**.

**✅ Final Thoughts**

This project demonstrates a **lightweight, real-time, ML-based IDS** using public datasets, with full end-to-end flow from model training to deployment.

You're now ready to demo or expand this system further for production or academic use!