Real-Time DDoS Detection System with Machine Learning

This project is a complete, end-to-end application for detecting Denial-of-Service (DDoS) attacks in real-time by monitoring web server logs. It uses a custom-trained Random Forest machine learning model to distinguish between normal (benign) and attack traffic. The system features an interactive dashboard for visualizing traffic patterns and detection status.

Features

- **Real-Time Log Monitoring**: A background script continuously watches the web server's access log for new entries.
- **Custom Model Training**: The system generates its own training data based on simulated local traffic, creating a highly accurate model tailored to its specific environment.
- **Machine Learning Detection**: A Scikit-learn RandomForestClassifier analyzes traffic features (like request rate) to classify traffic as "Benign" or "DDoS".
- **Interactive Dashboard**: A Streamlit web application provides a live view of the system's status, traffic volume, top attacking IPs, and a breakdown of predictions.
- **Modular Architecture**: The application is broken into decoupled components: a web server, a monitoring service, a user interface, and an attack simulator, making it easy to understand and maintain.

System Architecture

The application operates using several independent components that communicate through log files and a central database.

1. **Flask Web Server (flask_server.py)**: A lightweight server that receives all HTTP traffic and writes every request to logs/access.log.
2. **Attack Simulator (ddos_test.py)**: A script to generate high-volume, multi-threaded HTTP requests to simulate a DDoS attack against the Flask server.
3. **Data Creation Script (create_training_data.py)**: A one-time script that processes a generated log file (containing both benign and attack traffic) and creates a labeled training_from_logs.csv file.
4. **Monitoring Service (monitoring.py)**: The core of the system. This standalone script:
    - Trains a machine learning model using the custom CSV file on its first run.
    - Continuously monitors access.log for new entries.
    - Calculates traffic features for incoming requests.
    - Uses the trained model to make predictions.
    - Stores all logs, features, and predictions in a SQLite database (logs/requests.db).
5. **Streamlit Dashboard (app.py)**: The front-end interface. It reads from the SQLite database to display the latest system status and visualizations.

Prerequisites

- Python 3.8+
- pip for package installation
- It is highly recommended to use a Python virtual environment (venv or conda).

Installation

1. **Clone the repository:**
   ```
   git clone <your-repository-url>
   cd <your-repository-directory>
   ```

2. **Create and activate a virtual environment:**
   ```
   # For Windows
   python -m venv venv
   .\venv\Scripts\activate

   # For macOS/Linux
   python3 -m venv venv
   source venv/bin/activate
   ```

3. **Install the required packages:**
   ```
   pip install -r requirements.txt
   ```

How to Run the System

Running the application is a two-phase process. First, you must generate data and train the model. After that, you can run the live system.

Phase 1: One-Time Setup (Data Generation & Model Training)

This phase creates the training_from_logs.csv file that the model will learn from.

1. **Start the Web Server**: Open your first terminal and run the Flask server.
   ```
   python flask_server.py
   ```

2. **Generate Traffic**:
   - **Benign Traffic**: Open a web browser and visit http://127.0.0.1:5000 a few times.
   - **Attack Traffic**: Open a second terminal, activate the virtual environment, and run the attack script for **15-20 seconds**, then stop it with Ctrl+C.
     ```
     python ddos_test.py
     ```

3. **Stop the Web Server**: Go back to your first terminal and stop the server with Ctrl+C.
4. **Create the Dataset**: In one of your terminals, run the data creation script.

```
python create_training_data.py
```

This will process logs/access.log and create data/training_from_logs.csv.

Phase 2: Running the Live System

For this phase, you will need **three separate terminals** running simultaneously.

1. **Terminal 1: Start the Flask Server**
   ```
   python flask_server.py
   ```
2. Terminal 2: Start the Monitoring Service
   This script will first train and save the model, then begin monitoring the log file.
   ```
   python monitoring.py
   ```
3. **Terminal 3: Start the Dashboard**
   ```
   streamlit run app.py
   ```
   A browser tab will open with the dashboard.
4. Terminal 4 (Optional): Simulate an Attack
   To test the live system, open a fourth terminal and run the attack script.
   ```
   python ddos_test.py
   ```
   Watch the monitor terminal for detection alerts and refresh the Streamlit dashboard to see the status change.

File Descriptions

- app.py: The Streamlit dashboard application.
- monitoring.py: The standalone service that monitors logs, computes features, and makes predictions.
- flask_server.py: The target web server that generates logs.
- model.py: Defines the DDoSModel class for training, loading, and predicting.
- database.py: Manages all interactions with the SQLite database.
- utils.py: Contains the compute_features helper function.
- visualization.py: Contains the functions for generating Plotly charts.
- create_training_data.py: Script to generate the custom training CSV.
- ddos_test.py: The attack simulation script.
- requirements.txt: A list of all required Python packages.