Step-by-Step Implementation for Automated Threat Detection System

Step 1: Requirement Analysis

- Define scope: Build a web-based anomaly detection system using ML for network admins.
- Requirements: Flask, Random Forest, Isolation Forest, CICIDS2017 dataset, Chart.js for visualization.
- Modules: Data Preprocessing, Feature Engineering, Model Training, Model Evaluation, API, Future Enhancement.

Step 2: Software Setup

- Install Python 3.8+, flask, numpy, pandas, sklearn, imblearn, cudf-cu11, cuml-cu11.
- Set up logging (atd.log).
- Store CICIDS2017 dataset in /mnt/c/atd/DATADIR.

Step 3: Data Preprocessing

- Load and merge CICIDS2017 CSV files.
- Clean data (remove NaN, infinities).
- Convert labels (0 for BENIGN, 1 for attacks).
- Scale features using StandardScaler.

Step 4: Feature Engineering

- Select 13 features (e.g., Flow Duration, Total Fwd Packets).
- Extract features into matrix X, labels into y.
- Normalize features with StandardScaler.

Step 5: Model Training

- Train or load models (iso_forest.pkl, rf_model.pkl, scaler.pkl):
    - Isolation Forest: contamination=0.05.

    - Random Forest: GPU-accelerated (cuml), n_estimators=2000.
- Balance classes with SMOTE.
- Save models as pickle files.

Step 6: Model Evaluation

- Evaluate Random Forest on test set (compute accuracy, log results).
- Log Isolation Forest predictions.
- Store metrics in atd.log.

Step 7: API Development

- Create Flask app with:
    - GET /: Serve dashboard.
    - POST /predict: Validate input, predict using models, return results.
- Add error handling and logging.

Step 8: UI Design

- Build dashboard (DASHBOARD_HTML):
    - Form for feature input.
    - Display predictions (Isolation Forest, Random Forest).
    - Charts (Confidence, History) using Chart.js.
- Style with CSS (Orbitron font, gradients).

Step 9: Testing

- Test data preprocessing, model training, API (/predict), and UI in browser.
- Debug using atd.log (e.g., model loading, input validation).

Step 10: Deployment

- Deploy on server with dataset and pickle files.
- Run Flask: python atd.py or use Gunicorn (gunicorn -w 4 -b 0.0.0.0:5000 atd:app).
- Ensure GPU support for cuml or switch to CPU.

Step 11: Evaluation and Future Enhancement

- Collect user feedback on usability, accuracy.
- Monitor atd.log for errors.
- Future enhancements: Real-time streaming, scalability, UI improvements.
- Document API, user guide, and include UML diagrams.