Network Threat Analysis System

A Machine Learning Pipeline for Network Intrusion Detection

GitHub: https://github.com/MoeMaloudi/Msc_Project/tree/Network_Behaviour_Analysis

1 What This Does

This system analyzes network traffic to detect cyber attacks. It processes PCAP files (network packet captures) and uses machine learning to identify different types of attacks like DDoS, port scans, web attacks, and more.

The system can handle large datasets and includes pre-trained models so you can start analyzing immediately.

1.1 Repository Access

GitHub Repository: https://github.com/MoeMaloudi/Msc_Project/tree/Network_Behaviour_Analysis

The repository includes:

- Complete Jupyter notebook with all processing cells
- Pre-trained models for immediate use (`Trained_Models_For_Quick_Demo/`)
- Pre-extracted features for ML training (`Features_For_Quick_Demo/`)
- Documentation and example outputs
- All necessary utility functions and helper scripts

2 Quick Start Options

2.1 Option 1: Use Pre-Trained Models (Fastest)

If you just want to test the system:

1. Open the notebook: `Network_Threat_Analysis_System_Parquet.ipynb`
2. **Run Cell 1 first** to import all required libraries
3. Run Cell 12 to load pre-trained models from `Trained_Models_For_Quick_Demo/`
4. Choose from: RandomForest, LightGBM, SGD, or XGBoost
5. Takes seconds to load, ready for immediate use

Expected Output:

- Model loading confirmation: "XGBoost model ready for use"
- Memory usage information
- Ready-to-use classifier for immediate network traffic analysis

2.2 Option 2: Train with Pre-Extracted Features

If you want to experiment with machine learning:

1. **Run Cell 1 first** to import all required libraries
2. Run Cell 13 to load features from `Features_For_Quick_Demo/`
3. Train your own models on pre-processed CICIDS2017 data
4. Takes 15-30 minutes
5. No need to download the 40 GB dataset or wait 2-3 hours for feature extraction

Expected Output:

- "Loaded 4,500,000 network flows" message
- Training progress with accuracy updates
- Model performance comparison results across different algorithms
- Feature importance rankings

Pre-extracted features include: Complete Tuesday-Friday CICIDS2017 data covering all 15 attack types including DDoS, PortScan, Web Attacks, Infiltration, and normal traffic.

2.3 Option 3: Full Pipeline

If you want to process your own PCAP files:

1. Get the CICIDS2017 dataset from https://www.unb.ca/cic/datasets/ids-2017.html
2. **Run Cell 1 first** to import all required libraries
3. Run all cells 1-11 for complete processing
4. Takes about 2-3 hours for the complete 40 GB dataset

Expected Output:

- Real-time processing statistics: "Processing rate: 1,247 flows/sec"
- Memory usage monitoring: "Memory usage: 1.8 GB / 8.0 GB"
- Feature extraction progress with attack detection counts
- Final trained models with comprehensive performance reports

3 What You Need

3.1 System Requirements

- 8-16 GB RAM
- Python 3.9+
- Jupyter Notebook
- 20+ GB free disk space (for full dataset)

3.2 Install Dependencies

Using conda (recommended):

Listing 1: Conda Installation

```
conda create -n network-analysis python=3.9
conda activate network-analysis
conda install pandas numpy scikit-learn matplotlib seaborn
conda install -c conda-forge lightgbm xgboost pyarrow plotly
pip install scapy pyshark ipywidgets psutil tqdm
```

Using pip:

Listing 2: Pip Installation

```
pip install pandas numpy scikit-learn matplotlib seaborn
pip install lightgbm xgboost pyarrow plotly ipywidgets
pip install scapy pyshark psutil tqdm
```

4 Repository Structure

```
Network-Analysis-Pipeline/
         Network_Threat_Analysis_System_Parquet.ipynb  # Main notebook
         Trained_Models_For_Quick_Demo/                # Ready-to-use models
               RandomForest/
               LightGBM/
               SGD/
               XGBoost/
         Features_For_Quick_Demo/                      # Pre-extracted features
             batch_parquet_files/
             compressed_features_csv/
```

5 Complete Cell Reference Guide

5.1 Core Pipeline Cells

Cell 1: Library Imports

- **Purpose:** Import all required Python libraries
- **Required for:** All other cells
- **Execution time:** 10-15 seconds
- **Output:** "All libraries imported successfully"

Cell 2: System Configuration

- **Purpose:** Interactive configuration UI for memory, paths, and processing settings
- **Required for:** Full pipeline (Cell 11)
- **Execution time:** Immediate (user interface)
- **Output:** Configuration widgets and settings validation

Cell 11: Main Pipeline Execution

- **Purpose:** Complete end-to-end processing of PCAP files and training
- **Dependencies:** Cells 1, 2
- **Execution time:** 2-3 hours for full CICIDS2017 dataset
- **Output:** Trained models, performance reports, visualizations

5.2 Quick Demo Cells

Cell 12: Load Pre-Trained Models

- **Purpose:** Load ready-to-use models for immediate inference
- **Dependencies:** Cell 1
- **Execution time:** Seconds
- **Output:** "XGBoost model ready for use" + model objects

Cell 13: Train with Pre-Extracted Features

- **Purpose:** ML training using pre-processed CICIDS2017 features
- **Dependencies:** Cell 1
- **Execution time:** 15-30 minutes
- **Output:** "Loaded 4,500,000 network flows" + training results

5.3 Optional Utility Cells

Cell 2.5: Create Test Dataset

- **Purpose:** Generate small dataset for quick testing
- **Dependencies:** Cell 1
- **Execution time:** 2-5 minutes
- **Output:** "Created test dataset with X packets"

Cell 14: CSV to Parquet Conversion

- **Purpose:** Convert CSV.gz feature files to Parquet format
- **Dependencies:** Cell 1
- **Execution time:** 2-5 minutes
- **Output:** "Successfully converted X files to Parquet format"

Cell 15: Generate Evaluation Reports

- **Purpose:** Create comprehensive reports from existing ML results
- **Dependencies:** Cell 1, saved ML results
- **Execution time:** 2-5 minutes
- **Output:** Interactive UI + generated report files

5.4 Execution Pathways

Full Pipeline: Cell 1 → Cell 2 → Cell 11 (2-3 hours)
Pre-trained Models: Cell 1 → Cell 12 (seconds)
Feature Training: Cell 1 → Cell 13 (15-30 minutes)
Testing: Cell 1 → Cell 2.5 → Cell 11 (minutes)
Report Generation: Cell 1 → Cell 15 (minutes)

6 How to Use

6.1 Main Notebook Cells

Required Cells:

- **Cell 1:** Import libraries
- **Cell 2:** Configuration
- **Cell 11:** Main processing (if using full pipeline)
- **Cell 12:** Load pre-trained models (quick demo)
- **Cell 13:** Train with pre-extracted features

Optional Utility Cells:

- **Cell 2.5:** Create small test dataset
- **Cell 14:** Convert CSV to Parquet format
- **Cell 15:** Generate evaluation reports from existing results

6.2 Optional Utility Cells Explained

6.2.1 Cell 2.5: Create Small Test Dataset

Purpose: Creates a smaller subset of data for quick testing and development

When to use:

- Testing the pipeline without waiting hours
- Development and debugging
- Learning how the system works
- Validating setup before full processing

What it does:

- Extracts first N packets from PCAP files
- Creates corresponding subset of CSV labels
- Processes in minutes instead of hours
- Validates entire pipeline functionality

Expected Output:

- "Created test dataset with X packets"
- Smaller PCAP and CSV files for testing
- Quick validation that everything works

6.2.2 Cell 14: Convert CSV to Parquet Format

Purpose: Converts compressed CSV feature files to Parquet format for faster ML processing

When to use:

- You have CSV.gz feature files but need Parquet format
- Converting your own extracted features for Cell 13
- Optimizing file format for faster loading

- Working with legacy feature extractions

What it does:

- Reads compressed CSV feature files
- Converts to Parquet format with compression
- Creates batch files suitable for Cell 13
- Improves loading speed for ML training

Expected Output:

- "Converting CSV files to Parquet format..."
- "Converted: features_chunk_000.csv.gz -> batch_0000.parquet"
- "Successfully converted X files to Parquet format"
- Ready-to-use Parquet files for Cell 13

Note: Skip Cell 14 if you're using the pre-made Parquet files in `Features_For_Quick_Demo/batch_parquet_`

6.2.3 Cell 15: Generate Evaluation Reports from Existing Results

Purpose: Create comprehensive analysis reports and visualizations from previously saved ML results without retraining

When to use:

- After completing training runs to generate detailed reports
- Creating publication-quality figures and analysis
- Regenerating reports in different formats
- Performing deeper analysis on existing results
- Creating thesis-ready outputs and summary tables

What it does:

- Loads saved ML results from JSON, Pickle, or CSV files
- Generates comprehensive text reports with detailed analysis
- Creates confusion matrix visualizations for all models
- Produces summary statistics in CSV format
- Supports multiple file formats and flexible data loading
- Provides interactive UI for easy file upload and configuration

Supported Input Files:

- **Metrics Files:** JSON, Pickle (.pkl), CSV containing model performance
- **Feature Files:** Pickle (.pkl), CSV, or TXT files with feature names
- **Confusion Matrices:** JSON or Pickle files with confusion matrix data

Generated Outputs:

- **Comprehensive Report:** Detailed text analysis with model comparisons
- **Confusion Matrices:** High-resolution PNG visualizations for each model
- **Summary Statistics:** CSV files with all performance metrics

- **Comparative Analysis:** Model ranking and performance tables

Expected Output:

- Interactive file upload interface with progress indicators
- "Files loaded successfully! Ready to generate reports."
- Generated report files in specified output directory
- "Successfully generated X report(s)" confirmation message

6.3 Using Pre-Trained Models

Listing 3: Loading Pre-Trained Models in Cell 12

```
# In Cell 12
import pickle

# Choose algorithm: "RandomForest", "LightGBM", "SGD", "XGBoost"
algorithm = "XGBoost"
model_path = f'./Trained_Models_For_Quick_Demo/{algorithm}/'

# Load model
with open(f'{model_path}xgboost_incremental.pkl', 'rb') as f:
    model = pickle.load(f)

# Load scaler and features
with open(f'{model_path}scaler.pkl', 'rb') as f:
    scaler = pickle.load(f)

print(f"{algorithm} model ready for use")
```

6.4 Using Pre-Extracted Features

Listing 4: Loading Pre-Extracted Features in Cell 13

```
# In Cell 13
import pandas as pd
import glob

# Load all feature batches
batch_path = './Features_For_Quick_Demo/batch_parquet_files/'
batch_files = glob.glob(f'{batch_path}batch_*.parquet')
features = pd.concat([pd.read_parquet(f) for f in batch_files])

print(f"Loaded {len(features)} network flows")
print("Ready for ML training")
```

7 What the System Detects

The system can identify these attack types:

- Normal traffic (BENIGN)
- DDoS attacks
- Port scanning
- Botnet activity
- Web attacks (SQL injection, XSS, brute force)

- DoS attacks (various types)
- Brute force attacks (FTP, SSH)
- Heartbleed exploit
- Network infiltration

8 Performance

- Processes 4.5 million network flows
- XGBoost gives best results
- Uses 8GB RAM with streaming processing
- Takes about 2 hours for complete pipeline
- Pre-trained models work immediately

9 Troubleshooting

Memory errors: Reduce memory settings in Cell 2 configuration
Missing files: Make sure PCAP and CSV files are in correct directories
Slow performance: Use pre-extracted features instead of full pipeline
Installation issues: Use conda instead of pip for better package management

10 CICIDS2017 Dataset

If using the full pipeline, you need the CICIDS2017 dataset:

- Download from: https://www.unb.ca/cic/datasets/ids-2017.html
- Size: About 40 GB total
- Files needed: Tuesday-Friday PCAP files and CSV labels
- May require institutional access

10.1 Pre-Extracted Features Source Data

The pre-extracted features in `Features_For_Quick_Demo/` were processed from these specific CICIDS2017 files:

PCAP Files (4 days):

- Tuesday-WorkingHours.pcap
- Wednesday-workingHours.pcap
- Thursday-WorkingHours.pcap
- Friday-WorkingHours.pcap

Ground Truth CSV Files (7 files):

- Tuesday-WorkingHours.pcap_ISCX.csv
- Wednesday-workingHours.pcap_ISCX.csv
- Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv
- Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv
- Friday-WorkingHours-Morning.pcap_ISCX.csv

- Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv
- Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv

This represents the complete Tuesday-Friday CICIDS2017 dataset covering all 15 attack types and normal traffic patterns.

File structure needed:

```
./data/CICIDS2017/Raw_PCAP/
         Tuesday-WorkingHours.pcap
         Wednesday-workingHours.pcap
         Thursday-WorkingHours.pcap
         Friday-WorkingHours.pcap

./data/CICIDS2017/TrafficLabelling/
         Tuesday-WorkingHours.pcap_ISCX.csv
         Wednesday-workingHours.pcap_ISCX.csv
         [additional CSV files for Thursday/Friday]
```

Note: This system is for educational and research purposes. Make sure you have permission to analyze network traffic data.