Network Intrusion Detection System (NIDS)

pwnFAST

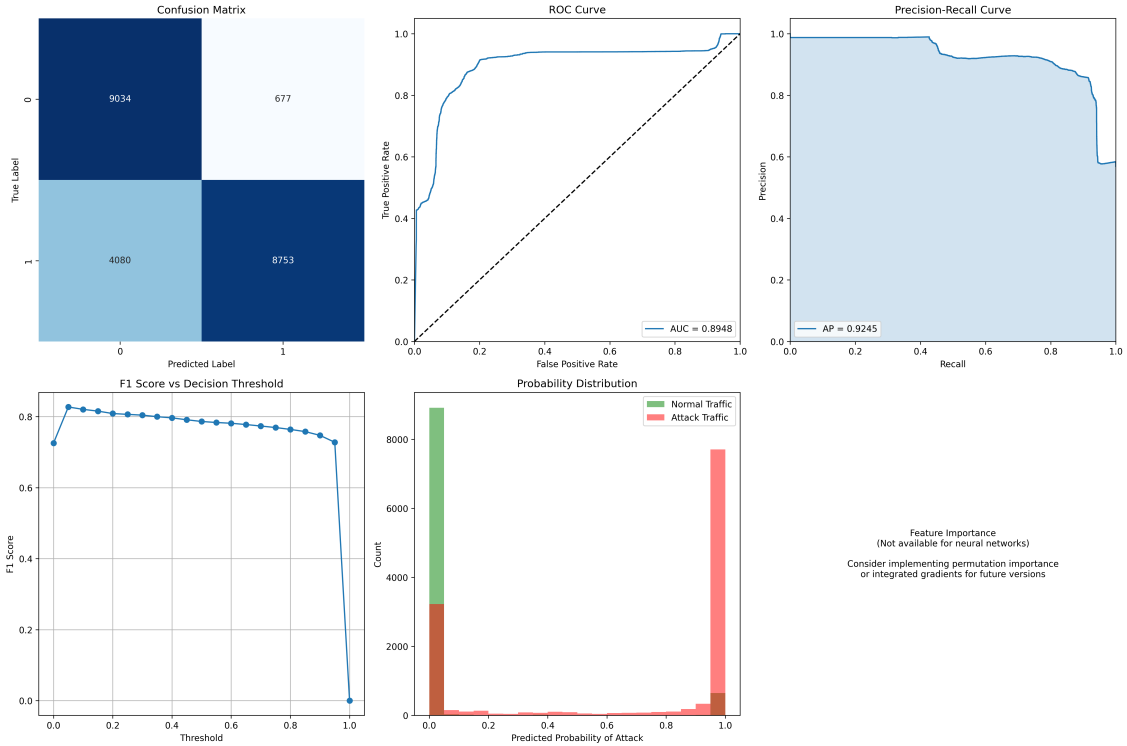
Department of Computer Science
FAST NUCES Karachi

April 30, 2025

Project Overview

Objective

Develop a real-time, machine learning-based NIDS trained on the NSL-KDD dataset to detect and alert on anomalous network traffic.



Key Features

- Real-time packet capture and analysis (Scapy)
- Deep learning model with high precision and balanced recall
- Detailed attack fingerprinting (port scans, stealth scans, DoS, ICMP)
- Configurable thresholds for false positive/negative trade-offs
- Comprehensive logging with contextual packet information

Model Performance

Metric	Score
Accuracy	79.28%
Precision	92.17%
Recall	69.50%
F1 Score	79.24%
AUC-ROC	86.51%

Confusion Matrix (Threshold = 0.10):

- True Positives: 8,919
- False Positives: 758
- True Negatives: 8,953
- False Negatives: 3,914

Threshold Recommendations

- **Balanced (0.10)**: FPR 7.81%, FNR 30.50%
- **High Security (0.05)**: lowers FNR further (to 20%), FPR 12%
- **Low False Positive (0.90)**: FPR 2.78%, FNR 34.70%

System Architecture

- Training Pipeline (PyTorch): data download, preprocess, model training
- Detection Engine: packet capture with Scapy, feature extraction
- Real-time Inference: anomaly scoring
- Alert Manager: contextual logging and notifications

Technical Implementation

Model Architecture:

```
ImprovedNIDSModel(
  (net): Sequential(
    Linear(122,128), ReLU(), BatchNorm1d(128), Dropout
        (0.3),
    Linear(128,64), ReLU(), BatchNorm1d(64), Dropout
        (0.3),
    Linear(64,32), ReLU(), BatchNorm1d(32), Dropout
        (0.15),
    Linear(32,1)
  )
)
```

Training Techniques:

- Weighted sampling for class imbalance
- Early stopping on validation F1
- Learning rate scheduling
- BCEWithLogitsLoss with class weights

Usage Instructions

Prerequisites:

```
pip install scapy pandas numpy scikit-learn torch
    torchvision joblib
```

Train Model:

```
python3 train_nids_model.py
```

Run Detection:

```
sudo $(which python3) run_nids.py
```

Future Improvements

- Deep packet inspection for application-layer threats
- Ensemble of ML models for robustness
- Dynamic thresholding based on network baseline
- Alert clustering to reduce noise
- Automated mitigation actions

Conclusion

Takeaways

A balanced, ML-driven NIDS can deliver real-time protection with high detection and manageable false positives.

Questions or Collaboration Ideas?