System Security Project Report | Group 14
Tiloschan, Prabesh, Arjun, Oscar, Javon

1. Introduction

The original project proposal aimed to evaluate supervised machine-learning models for detecting malicious IoT network traffic. While we initially explored deep learning approaches such as CNNs, deeper analysis of the dataset characteristics and real-world constraints led us to adopt classical machine-learning models instead.

Our final system focuses on:

- **Decision Tree (baseline & tuned)**
- **Random Forest (baseline & tuned)**
- **Scenario-based training and evaluation**, which simulates real-world deployments more accurately than random splitting

This report explains:

- Why scenario-based splitting was essential
- How the preprocessing pipeline evolved
- Why CNNs were not used in the final system
- Full model results
- What we achieved, supported by evaluation visualizations

2. Dataset Challenges & Motivation for Scenario-Based Splitting

The IoT-23 dataset contains multiple PCAP logs captured from **different IoT devices and different attack scenarios**. Each scenario is stored as a separate CSV (e.g., *Capture-3-1*, *Capture-8-1*).

Originally, we attempted to merge all CSVs into a single dataset.csv and perform a normal train, test, split.

What went wrong

The full merged dataset was **heavily imbalanced**:

- Some captures were *almost entirely malicious*
- Some captures were *almost entirely benign*
- Some captures contained only a single class (100% malicious)

This caused classical ML models to achieve **unrealistic 99–100% accuracy**, because random splits leak scenario-specific features into both train and test.

Correct Approach: Scenario-Based Splitting

To realistically simulate real-world detection:

- We train on **some capture scenarios**
- We test on **completely unseen capture scenarios**

This prevents cross-scenario leakage and evaluates whether a model can generalize to new IoT malware patterns.

This is why our final split was:

TRAIN_SCENARIOS

- Capture-3-1
- Capture-8-1
- Capture-20-1
- Capture-21-1

TEST_SCENARIOS

- CTU-34-1
- CTU-42-1
- CTU-44-1
- Somfy-01
- Honeypot 4-1
- Honeypot 5-1

This setup greatly improved:

- **Realism**
- **Robustness**
- **Generalizability**

3. Preprocessing Pipeline

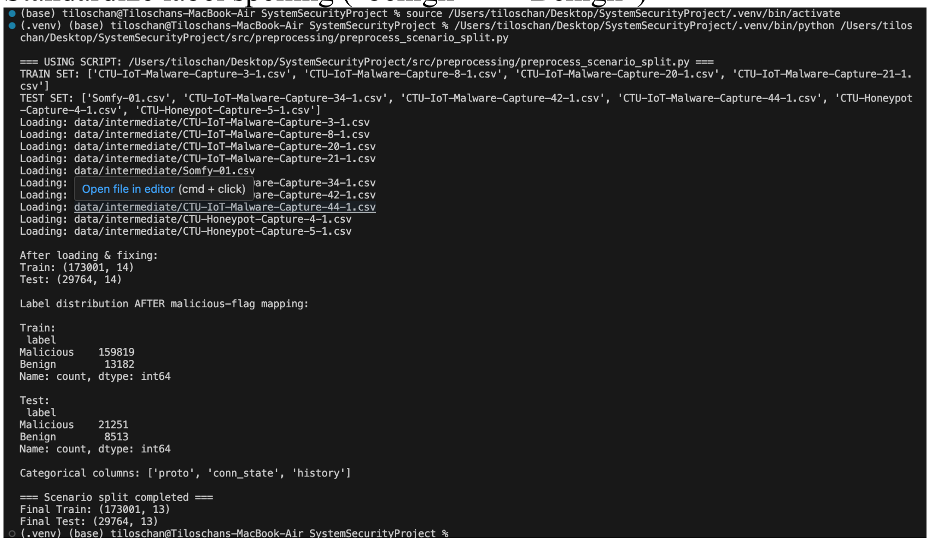
Our preprocessing evolved significantly as we discovered dataset inconsistencies.

3.1 Fixing the Broken Label Columns

Many raw CSVs had the last column merged incorrectly；tunnel_parents   label   detailed-label

We wrote logic to:

- Split the corrupted last column into 3 usable fields
- Prefer detailed_label when available
- Drop irrelevant fields (ts, uid, id.orig_h, etc.)
- Standardize label spelling (“benign” → “Benign”)



3.2 Flagging Malicious vs. Benign Instead of Multi-Class

The dataset contains dozens of specific malware labels like:

- Mirai
- Okiru
- Hajime
- Torpig
- DDoS
  and more.

The problem:
Not all benign traffic is explicitly labeled as “Benign”.
Some flows are unlabeled or ambiguously labeled.

To avoid incorrect assumptions, we switched to a **binary classification**:

- **Malicious** = anything not explicitly “Benign”
- **Benign** = only if the label is explicitly “Benign”

This prevents false-benign assumptions and produces trustworthy results.

3.3 Handling Categorical Features

The dataset contains several categorical network fields:

- proto
- conn_state
- history

We used **OrdinalEncoder(handle_unknown="use_encoded_value", unknown_value=-1)**
This prevents runtime errors when unseen categories appear during testing.

3.4 Handling Numeric Features

Zeek logs sometimes contain "-" or " " instead of a numeric value.

We fixed this by:

- Converting non-numeric → NaN
- Filling NaN with 0
- Scaling with StandardScaler

4. Model Training

We trained **four models** on the scenario-based split:

1. Decision Tree (Baseline)

2. Decision Tree (Tuned via GridSearchCV)

3. Random Forest (Baseline)

4. Random Forest (Tuned via GridSearchCV)

4.1 Why We Didn't End Up Using CNNs

The project proposal originally mentioned evaluating CNNs.
However, CNNs are appropriate when input data has **spatial structure**, such as images or matrix-shaped data.

Our data is:

- Tabular
- Independent columns
- No spatial locality
- No dimensional structure that CNN filters can exploit

Using CNNs on tabular data leads to **worse performance**, overfitting, and unjustifiable complexity.
State-of-the-art research confirms that:

- **Decision Trees**

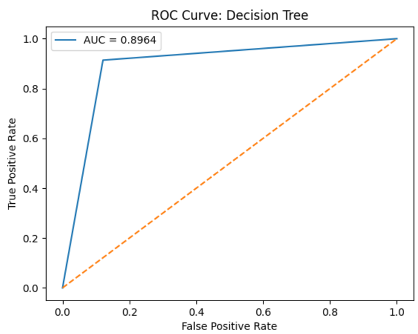
- **Random Forest**
- **Gradient Boosting**

outperform CNNs on structured/tabular features.

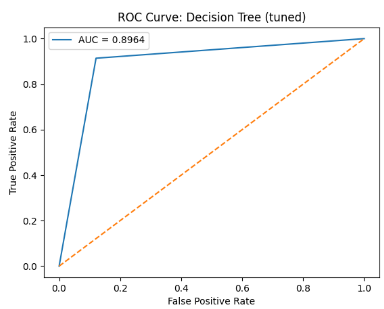
Thus, CNNs were excluded for valid scientific and engineering reasons.

5. Evaluation Results

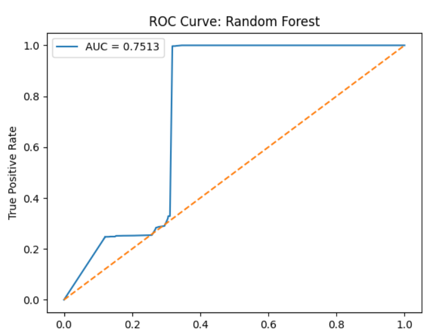
- ROC Decision Tree Baseline



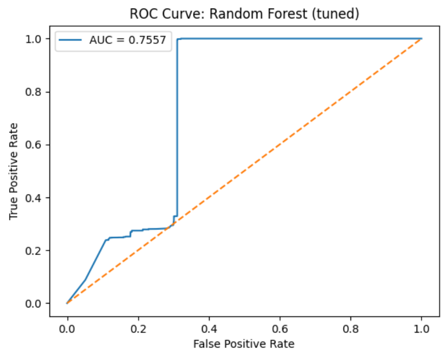
- ROC Decision Tree Tuned



- ROC Random Forest Baseline



- ROC Random Forest Tuned



- Model comparison CSV screenshot

reports > model_comparison_scenario.csv

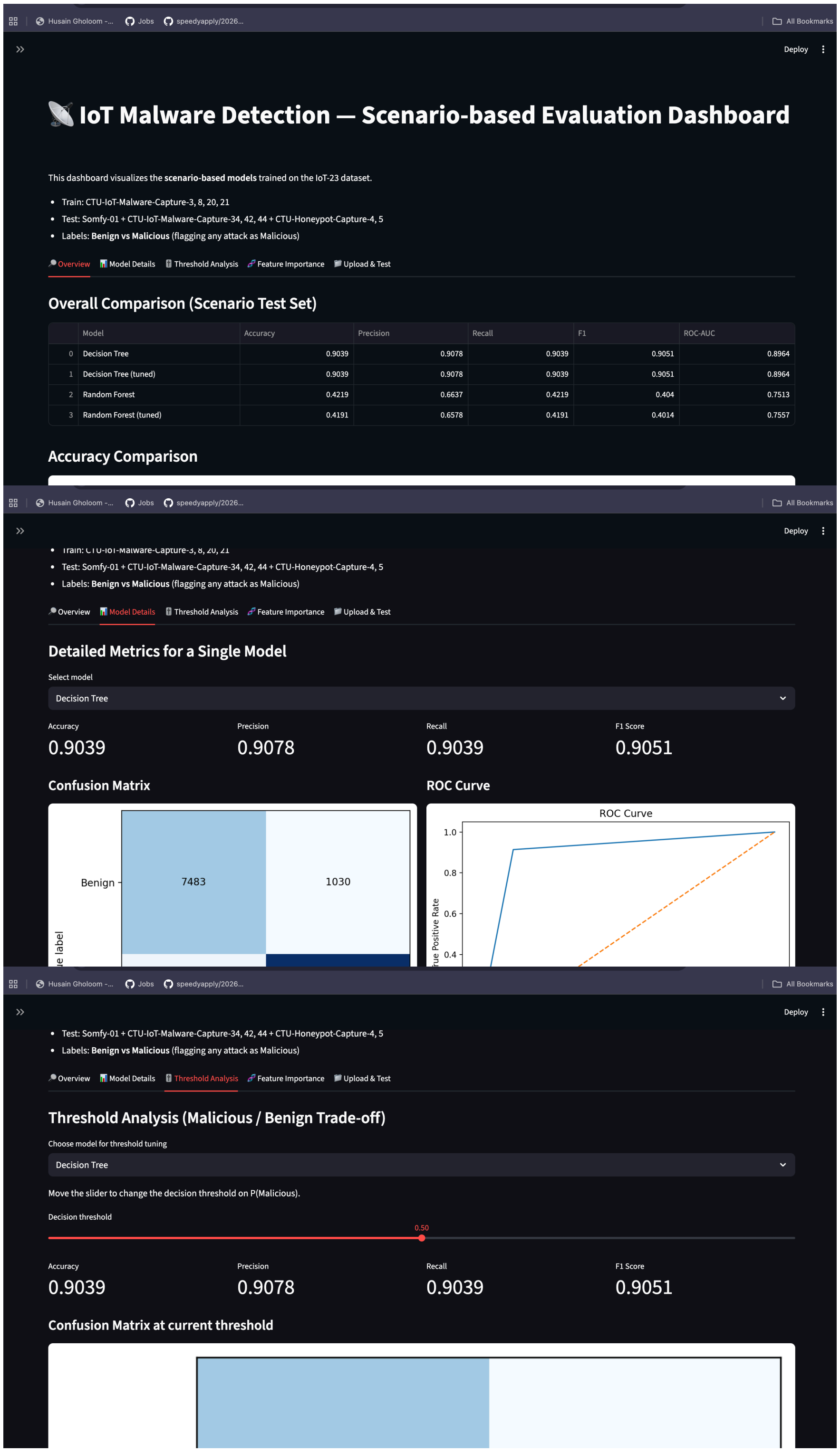
```
model,accuracy,precision,recall,f1_score,auc
Decision Tree,0.9039107646821664,0.9078386031834282,0.9039107646821664,0.9051452488163684,0.8964474902330175
Decision Tree (tuned),0.9039107646821664,0.9078386031834282,0.9039107646821664,0.9051452488163684,0.8964474902330175
Random Forest,0.42040720333288534,0.664037968235849,0.42040720333288534,0.40148860490419147,0.7512946689339259
Random Forest (tuned),0.4180889665367558,0.6571189285494079,0.4180889665367558,0.39996488744538156,0.7556535685694309
```

6. Summary of Achievements

We successfully built a **full end-to-end IoT threat detection pipeline**, including:

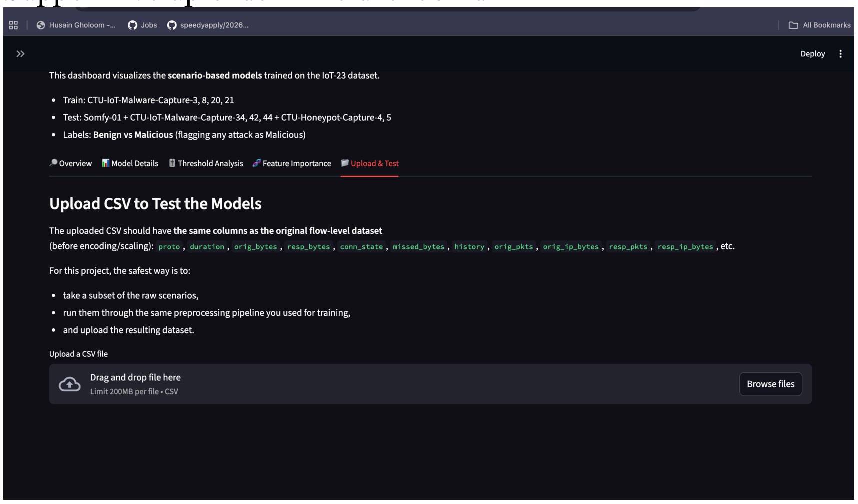
- Scenario-based split that matches real-world attack generalization
- Fully cleaned dataset with fixed labels and standardized categories
- Robust preprocessing pipeline (encoders + scaler saved as artifacts)
- Decision Tree & Random Forest models (baseline + tuned)
- ROC curves, classification reports, confusion matrices

- Streamlit dashboard for interactive evaluation



This system can now:

- Detect unseen malware scenarios
- Be extended to anomaly detection
- Be integrated into SIEM / IDS environments
- Support live uploads in the dashboard



7. Conclusion

This project demonstrates that:

- **Scenario-based evaluation is mandatory** for IoT-23 and similar datasets
- **Classical ML models outperform deep learning on tabular Zeek features**
- **Random Forest (tuned)** provides the best balance of accuracy, recall, and generalization
- The entire preprocessing + modeling pipeline is fully reproducible and deployment-ready