ML-IDS

Thomas Escaffre Azhar El Hidaou Illan Coco Guignard

November 2024

Intelligent IoT Intrusion Detection System (ML-IDS)

Introduction

With the proliferation of Internet of Things (IoT) devices, smart homes have become increasingly interconnected but also more vulnerable to cyberattacks. Attackers exploit these vulnerabilities to infiltrate networks, compromise devices, and steal sensitive data. To address this, our project develops a **Machine Learning-based Intrusion Detection System (IDS)** capable of analyzing network traffic in real time to detect anomalies and prevent intrusions.

The primary goal is not only to achieve high prediction accuracy but also to ensure the interpretability of the model, which is crucial for identifying patterns indicative of attacks. This document outlines the dataset, methods, and results while explaining the rationale and impact of each step in detail.

Why This Dataset?

The dataset chosen for this project is ideal for several reasons:

- **Diverse Features:** It contains 24 attributes representing various aspects of network traffic, including packet size, protocol type, service type, and connection statistics. This diversity allows the model to learn a comprehensive representation of network behavior.
- **Balanced Classes:** The dataset is evenly split between normal and attack instances, avoiding the pitfalls of class imbalance that can bias the model toward majority classes.
- **Practical Relevance:** The features in this dataset are typical of real-world network traffic, making the results more generalizable to actual IoT environments.
- **Challenge for Optimization:** The large number of features, some highly correlated, makes this dataset a good test case for feature engineering techniques like Recursive Feature Elimination (RFE).

Project Objectives

1. **Data Preparation and Cleaning:** Ensure that the dataset is ready for analysis by encoding categorical variables, normalizing numerical data, and removing irrelevant or redundant information.
2. **Feature Selection and Dimensionality Reduction:** Optimize the dataset to improve model performance and interpretability without sacrificing critical information.
3. **Model Training and Evaluation:** Experiment with baseline models (Logistic Regression and Decision Tree), assess their performance, and refine them using selected features.
4. **Comprehensive Analysis:** Use metrics and visualizations (confusion matrices, ROC curves) to understand and validate the models' performance in the context of intrusion detection.
5. **Model Interpretability:** Prioritize clarity in understanding what features contribute most to identifying attacks, ensuring practical utility in cybersecurity contexts.

Detailed Step-by-Step Process

1. Data Loading and Exploration

The dataset was loaded into a DataFrame, and its structure was examined using methods like `info()` and `describe()`. Key observations included:

- **No Missing Values:** All features were fully populated, simplifying the preprocessing stage.
- **Variety of Data Types:** Some columns contained numerical data (e.g., `src_bytes`, `dst_bytes`), while others were categorical (e.g., `protocol_type`, `attack`).
- **Large Scale Variations:** Numerical columns like `src_bytes` and `dst_bytes` showed ranges from 0 to over a billion, necessitating normalization.

2. Data Cleaning

- **Encoding Categorical Variables:** Columns like `protocol_type` and `attack` were converted into numerical formats using `LabelEncoder`. For example:
  - The `attack` column was transformed from `Yes/No` to `1/0`, enabling binary classification.

- **Outlier Removal:** Using Z-Score filtering, rows with extreme values were removed to prevent skewing the model. However, the threshold (4) was carefully chosen to retain significant outliers indicative of specific attacks, such as brute-force attempts.

Impact: This step made the dataset compatible with machine learning algorithms while preserving critical attack data.

3. Data Normalization

The dataset's numerical columns had varying scales, which could lead to bias in model training. For example:

- **`src_bytes` and `dst_bytes`:** Large-scale features dominate smaller ones like `duration`.

Using `StandardScaler`, all numerical features were normalized to have a mean of 0 and a standard deviation of 1.

Impact: Normalization ensured that all features contributed equally to the model, improving stability and convergence during training.

4. Feature Engineering and Correlation Analysis

- **Correlation Matrix:** The heatmap provided valuable insights:
  - Strong positive correlation between `srv_count` and `same_srv_rate` suggested redundancy.
  - Features like `dst_host_same_srv_rate` correlated moderately with `attack`, indicating their importance for classification.
- **Feature Selection with RFE:** RFE identified the top 10 most predictive features, such as: `protocol_type`, `logged_in`, `count`, `serror_rate`, and `rerror_rate`.

Impact: Reducing the feature set simplified the model without significant loss of information, making it faster and easier to interpret.

5. Model Development

- **Baseline Models:** Logistic Regression and Decision Trees were chosen for their simplicity and interpretability.
  - Logistic Regression relies on linear relationships between features and the target.
  - Decision Trees provide a visual representation of decision-making criteria.

- **Training and Testing Split:** The dataset was split into 80% training and 20% testing sets. Stratification ensured balanced class distributions across splits.
- **RFE Comparison:** Models were trained both with the full dataset and the RFE-reduced dataset to evaluate the impact of feature selection.

Impact: These baseline models served as a foundation for understanding the dataset and identifying potential improvements.

6. Model Evaluation and Visualization

Metrics like **Accuracy, Precision, Recall, F1-Score,** and **Confusion Matrices** were used to evaluate the models:

- **Logistic Regression:**
  - Accuracy: 93.2% without RFE; 91.4% with RFE.
  - Strengths: Robust and efficient, even with all features.
- **Decision Tree:**
  - Accuracy: 94.5% without RFE; 90.3% with RFE.
  - Strengths: Highly interpretable; identified key features like `src_bytes`.
- **ROC Curves:** Both models exhibited AUC values close to 1, confirming their strong discriminative ability.

Impact: These evaluations demonstrated the models' reliability and identified opportunities for refinement, such as tuning hyperparameters or exploring ensemble methods.

Key Results

- The models achieved high accuracy, precision, and recall, with minimal misclassifications.
- Decision Trees provided actionable insights into attack patterns, essential for cybersecurity applications.
- RFE highlighted critical features but slightly reduced model accuracy, suggesting a need for further refinement.

Conclusions

This project successfully established a foundational IDS model:

- **Data Cleaning:** Minimal adjustments were needed due to the dataset's quality.
- **Feature Selection:** RFE and correlation analysis aligned well with model performance.
- **Baseline Models:** Both Logistic Regression and Decision Trees performed exceptionally, validating the approach.

Future Directions

1. **Advanced Models:** Explore Random Forests or Gradient Boosting for improved accuracy. Explore Deep learning and Ensemble Learning.