GitHub link

https://github.com/Harish-kamal2005

Detecting Cyber Threats through Anomaly Detection in Network Traffic Data

1. Problem Statement

Detecting cyber threats in network traffic is crucial for ensuring network security. Anomaly detection methods analyze network traffic patterns to identify suspicious activities. These techniques help organizations mitigate risks and safeguard their assets from potential cyberattacks.

2. Project Objectives

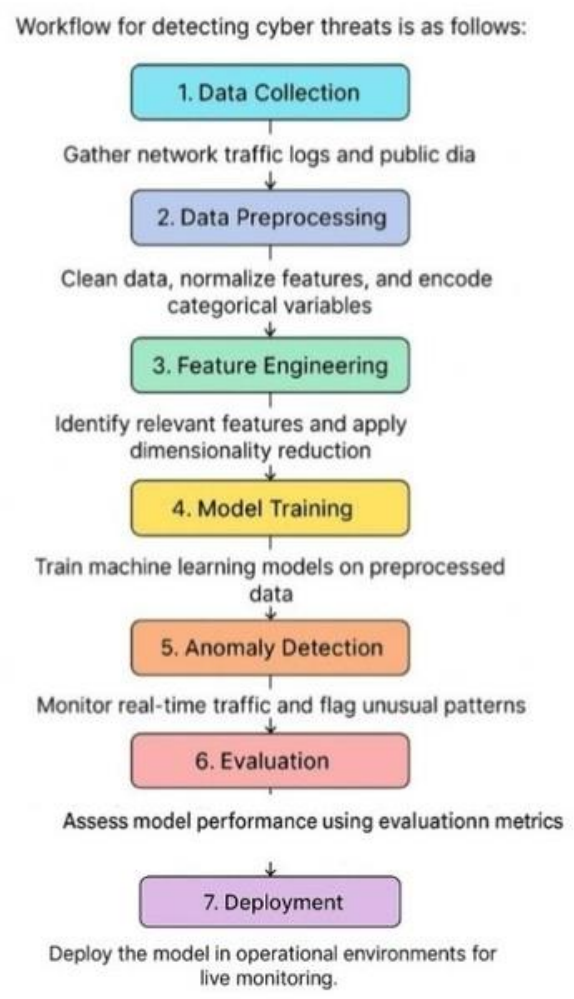
- Develop an anomaly detection model to identify potential cyber threats.

- Analyze network traffic data to understand patterns of normal and abnormal behavior.

- Provide actionable insights for real-time threat detection.

- Enhance the interpretability and usability of the model in practical scenarios.

3. Flowchart of the Project Workflow


Workflow for detecting cyber threats is as follows:
1. Data Collection
Gather network traffic logs and public dia
2. Data Preprocessing
Clean data, normalize features, and encode categorical variables
3. Feature Engineering
Identify relevant features and apply dimensionality reduction
4. Model Training
Train machine learning models on preprocessed data
5. Anomaly Detection
Monitor real-time traffic and flag unusual patterns
6. Evaluation
Assess model performance using evaluationn metrics
7. Deployment
Deploy the model in operational environments for live monitoring.

4. Data Description

The dataset contains 9,537 records and 11 features:

- Attributes: network_packet_size, protocol_type, login_attempts, etc.

- Target: attack_detected (0 for no attack, 1 for an attack).

This static dataset includes numerical and categorical variables relevant to network traffic analysis.

5. Data Preprocessing

- Checked and confirmed data integrity (no missing or null values).

- One-hot encoded categorical variables (e.g., protocol_type).

- Scaled numerical features using Min-Max scaling.

- Handled outliers by identifying anomalies using statistical methods.

6. Exploratory Data Analysis (EDA)

- Univariate Analysis:

 * Analyzed distribution of session_duration and login_attempts.

 * Observed frequency of attack_detected.

- Bivariate Analysis:

 * Examined correlations between network_packet_size and attack_detected.

 * Investigated relationships between ip_reputation_score and failed_logins.

Key Insights:

- Higher ip_reputation_score often correlates with suspicious activity.

- Frequent failed logins are strong indicators of potential threats.

7. Feature Engineering

- Created derived features (e.g., login_failure_rate = failed_logins / login_attempts).

- Encoded binary features for compatibility with machine learning models.

- Removed redundant or highly correlated features to reduce noise.

8. Model Building

- Used Random Forest and Isolation Forest for anomaly detection.

- Split data into 80% training and 20% testing subsets.

- Evaluated models using precision, recall, and F1-score.

- Fine-tuned hyperparameters to optimize model performance.

9. Results & Insights

- Random Forest achieved a high precision score, effectively minimizing false positives.

- Isolation Forest successfully identified novel anomalies in the test data.

- Feature importance analysis highlighted ip_reputation_score and failed_logins as key predictors.

10. Tools and Technologies Used

- Python libraries: pandas, numpy, matplotlib, scikit-learn. .

- Notebook environment: Google Colab

- Data visualization: seaborn, plotly.

- Deployment: Flask for integrating models into an API.

11. Team members and roles

- Data Preparation and Cleaning:Adhithya.A
- Feature Engineering and EDA:Harisharan.p.s
- Anomaly Detection Modeling: Harish.k
- Results Interpretation and Reporting: Prasanth.p