Cybersecurity Threat Detection Algorithm using Machine Learning

By: VELAGALA KESAVA NAGA SAI ADEEP REDDY

Why I Chose This Algorithm:

I selected this algorithm because it provides a systematic and scalable approach to detect various types of cybersecurity threats using both rule-based and machine learning-based techniques.
By combining real-time data processing with anomaly detection and classification models, this algorithm enhances threat visibility and enables proactive responses in a large-scale environment.

Step 1: Data Collection

Collect logs and event data from:

- Network traffic (e.g., NetFlow, PCAP files)
- Firewalls, IDS/IPS logs (Snort, Suricata)
- System logs (e.g., syslog, Windows Event Viewer)
- Application logs (e.g., Apache, Nginx)

Tools: Apache Kafka, Flume, or Logstash for real-time streaming.

Step 2: Data Preprocessing

Convert raw logs into structured format (JSON, CSV).

Apply parsing, filtering, and normalization.

Handle missing values and outliers.

Feature extraction: IPs, ports, protocol types, bytes transferred, timestamps.

Flag known blacklisted IPs/domains.

Step 3: Feature Engineering

Encode categorical variables (e.g., One-hot encoding).

Generate time-based features (e.g., connection rate per IP).

Apply PCA and TF-IDF for dimensionality and text processing.

Step 4: Data Splitting

Split data into:

- Training Set (70%)

- Validation Set (15%)

- Test Set (15%)

Step 5: Model Selection

Choose models based on task:

- Binary classification: Random Forest, Logistic Regression, SVM

- Anomaly detection: Isolation Forest, One-Class SVM, Autoencoders

- Clustering: DBSCAN, K-Means

Step 6: Model Training

Use Scikit-learn, PySpark MLlib, or TensorFlow.

Tune hyperparameters using GridSearchCV or RandomizedSearchCV.

Evaluate with Accuracy, Precision, Recall, F1-Score, ROC-AUC.

Step 7: Model Evaluation

Test the model on unseen data.

Use a confusion matrix and k-fold cross-validation for performance check.

Step 8: Threat Labeling and Interpretation

Map predictions to threat categories like Malware, DDoS, etc.

Use SHAP or LIME for explaining model decisions.

Step 9: Real-Time Detection Pipeline

Deploy using Kafka + Spark Streaming.

Serve model via Flask/REST API.

Push alerts to dashboards like Kibana or Grafana.

Step 10: Continuous Learning & Feedback Loop

Retrain models with new data regularly.

Incorporate feedback to reduce false positives.

Update threat intel sources.