Real-Time Network Intrusion Detection System (NIDS) Using Apache Kafka, Apache Spark, and MLlib

---

1. Introduction

Network Intrusion Detection Systems (NIDS) are critical components in cybersecurity, designed to detect malicious or abnormal network traffic. With the rise of big data technologies, traditional IDS solutions are not efficient enough to process massive network traffic streams in real time.

This project implements a **Real-Time Intrusion Detection System** using: - **Apache Kafka** (Real-time data ingestion) - **Apache Spark Structured Streaming** (Streaming ETL + ML inference) - **Spark MLlib** (Model training & prediction) - **MongoDB / Cassandra** (Alerts storage) - **Visualization Layer** (Grafana or Web dashboard)

The system processes high-volume streaming network traffic, applies machine learning models to detect threats, and stores alerts for visualization and monitoring.

---

2. Problem Statement

Modern networks generate massive amounts of traffic. Detecting intrusions manually or using legacy batch-processing models is slow and ineffective. This project aims to:

- Detect attacks such as **DoS, DDoS, Brute Force, Botnet, Port Scans, and Web Attacks**.
- Build an end-to-end big data pipeline using real-time tools.
- Use machine learning to classify incoming packets as benign or malicious.
- Provide real-time dashboards for monitoring.

---

3. Dataset Used — CIC-IDS 2017

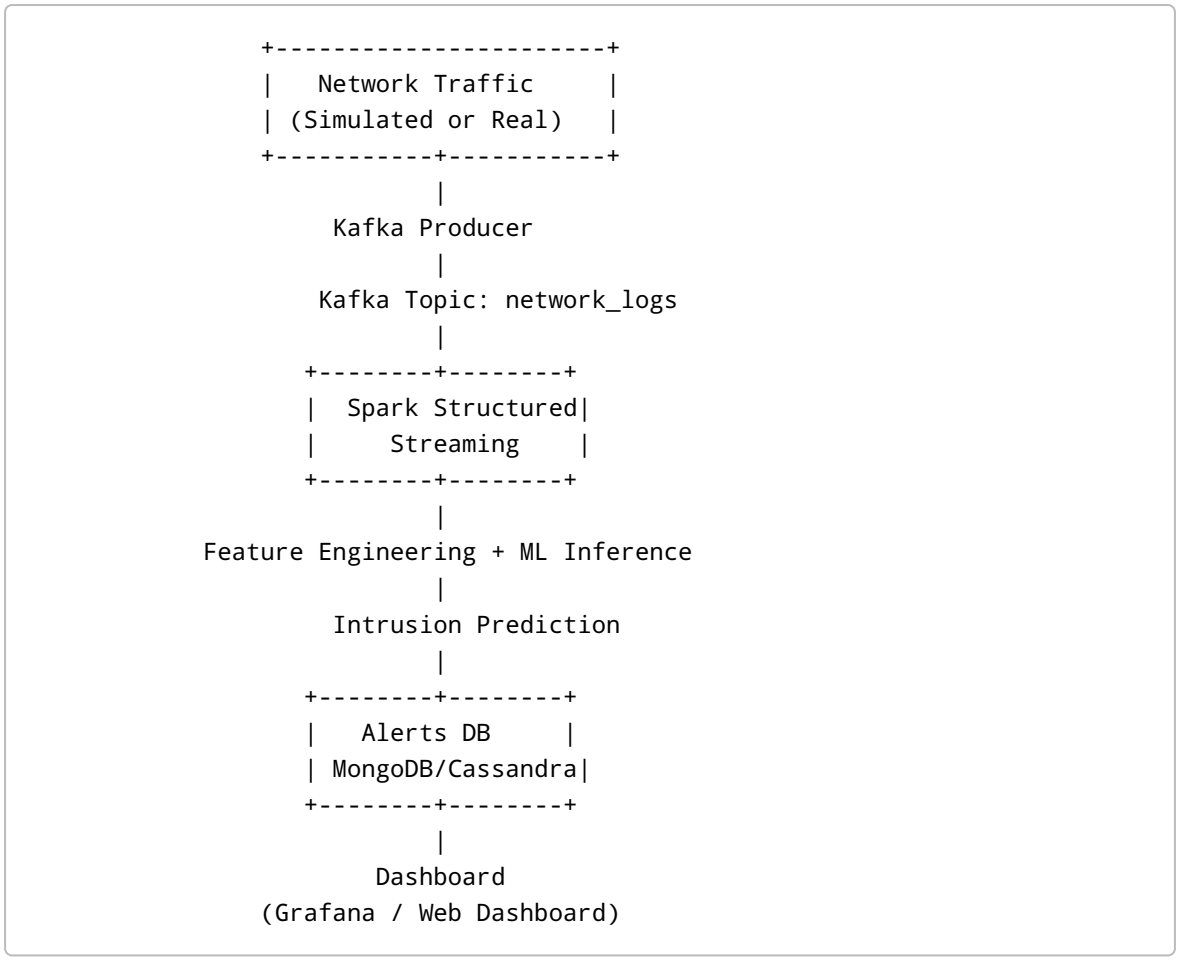
CIC-IDS 2017 is chosen because it: - Is modern and realistic - Contains 80+ network flow features - Includes multiple attack types - Is widely used in research - Works well with Spark for both batch and streaming

Each row in the dataset represents a network flow with features such as: - Flow Duration - Total Fwd Packets - Average Packet Size - Flow IAT Mean - Fwd Header Length - Subflow Fwd Bytes - Label (BENIGN or ATTACK TYPE)

---

4. System Architecture

The architecture consists of multiple interconnected layers.



---

5. Technologies Used

Big Data Frameworks

- Apache Kafka: Distributed messaging system
- Apache Spark: Distributed computation engine
- Spark Structured Streaming: Real-time streaming API

Machine Learning

- Spark MLlib
- Algorithms used:
- Random Forest Classifier
- Isolation Forest (Anomaly detection)

Storage

- MongoDB or Cassandra

- HDFS for batch storage

Visualization

- Grafana
- Plotly/Folium (optional)

---

6. Project Modules

6.1 Module 1 — Data Preprocessing (Batch Mode)

Tasks: - Handle missing values - Convert categorical features - Scale numerical values - Create training dataset

Tools used: - PySpark DataFrame API - MLlib Transformers

6.2 Module 2 — Model Training (Batch Mode)

Steps: 1. Load cleaned CIC-IDS dataset 2. Select features 3. Apply VectorAssembler 4. Train ML models 5. Save trained ML Pipeline model to disk

6.3 Module 3 — Kafka Producer (Real-Time)

The producer: - Reads rows from CIC dataset - Sends them to Kafka topic `network_logs` - Uses JSON format for Spark compatibility

6.4 Module 4 — Spark Streaming Job

Actions performed: - Reads JSON logs from Kafka - Applies same preprocessing transformers - Loads trained ML Pipeline model - Makes real-time predictions - Stores alerts in MongoDB/Cassandra

6.5 Module 5 — Dashboard

Visualizations include: - Intrusions per second - Attack types distribution - Geo-map of attacks (optional) - Real-time alert stream

---

7. ML Model Design

Model Options

1. **Random Forest (Classification)**
2. Good for tabular data
3. Handles non-linear patterns
4. **Isolation Forest (Anomaly Detection)**
5. Detects unseen attacks

6. Works well in streaming

Input Features

- 40–50 selected features from CIC-IDS
- Labels converted into binary: BENIGN vs ATTACK

Evaluation Metrics

- Accuracy
- Precision/Recall
- F1-score
- ROC-AUC

---

8. Pipeline Design (Step-by-Step)

Step 1: Download & Clean Dataset

Step 2: Preprocess with PySpark

Step 3: Train ML Model

Step 4: Save Pipeline

Step 5: Start Kafka Broker

Step 6: Run Kafka Producer

Step 7: Run Spark Streaming Job

Step 8: Insert Alerts into DB

Step 9: Show Dashboard

---

9. Sample Code (High-Level Overview)

(Code sections will be added after pipeline setup if needed)

---

10. Performance Optimizations

- Use Spark caching for frequent transforms
- Increase Kafka partitions for higher throughput
- Use checkpointing in structured streaming
- Enable Tungsten + Catalyst optimizer in Spark

---

11. Testing Strategy

1. Unit tests for preprocessing

2. Integration tests with Kafka

3. Model validation using test dataset

4. Stress test using large batches of packets

---

12. Deployment Strategy

Deployment can be done via: - Docker containers - Kubernetes cluster (optional) - Cloud platforms: AWS EMR, Databricks, GCP Dataproc

---

13. Results / Expected Outcomes

- Real-time detection of intrusions under 2 seconds
- Classification accuracy ~90% depending on model
- Live dashboard showing attack trends

---

14. Conclusion

This project demonstrates: - End-to-end big data pipeline creation - Real-time machine learning inference - Use of industry tools like Kafka, Spark, and MLlib - Cybersecurity application with real datasets

This makes it a strong, resume-worthy project suitable for academic and professional environments.

---

15. Future Enhancements

- Add deep learning (LSTM/Autoencoders) using PyTorch + Spark
- Real network packet capture using Wireshark + Kafka
- Multi-class attack classification
- Deploying in Kubernetes

---

16. References

- CIC-IDS 2017 Dataset
- Spark Documentation
- Kafka Documentation
- MLlib Guide