AI-Powered Blockchain Threat Detection and Real-Time Cybersecurity Alerts

Aim:

This project applies Artificial Intelligence (AI) to detect blockchain security threats, anomalies, and malicious activities in real time, and generates instant cybersecurity alerts to mitigate risks.

Dataset:

A synthetic and real-world blended dataset representing blockchain network activity, smart contract logs, and transaction anomalies.

- **Transaction Data:** Transaction hashes, block numbers, timestamps, gas usage, sender/receiver addresses.
- **Network Data:** Node communication logs, latency, packet drops, consensus delays.
- **Smart Contract Logs:** Function calls, gas patterns, abnormal state transitions.
- **Threat Labels:** Normal, phishing, reentrancy attack, Sybil attack, double-spending, DoS.
- **Alert Metadata:** Severity levels (Low–Critical), attack type classification, response actions.

Data characteristics include noisy inputs, imbalanced threat occurrences, and adversarial attack samples for robust model training.

Components:

1. **Blockchain Event Monitoring:** Continuous extraction of transaction and smart contract data.
2. **Threat Detection Module:** AI models (ML/DL) analyzing anomalies and malicious activity.
3. **Alert Engine:** Real-time classification of threat type and severity.
4. **Dashboard & Visualization:** Display live alerts, attack trends, and network health metrics.

5. **Integration Layer:** API/webhooks to notify security teams via email, SMS, or system logs.

Algorithms:

- **Isolation Forest / Autoencoders:** Detect anomalies in transaction patterns.
- **Random Forest / Gradient Boosting:** Classify known threats.
- **LSTM / RNN Models:** Capture temporal dependencies in blockchain activities.
- **Graph Neural Networks (GNNs):** Detect Sybil attacks and malicious clusters.

Evaluation Metrics:

- Accuracy, Precision, Recall, and F1-score for threat classification.
- False Positive Rate (FPR) – critical for alert systems.
- Detection Latency – average time to detect and alert.
- ROC-AUC Score – performance on imbalanced attack datasets.

Outcomes and Insights:

- Identifies and classifies blockchain threats with high accuracy.
- Provides **real-time alerts** to minimize damage from ongoing cyberattacks.
- Creates a transparent audit trail of security incidents for forensic analysis.
- Enhances blockchain trust and resilience by reducing vulnerabilities.

Methodology:

1. **Problem Definition:** Define blockchain-specific cyber threats (reentrancy, Sybil, double-spending).
2. **Dataset Collection & Preprocessing:** Aggregate blockchain transaction and log data, clean noise, handle missing values, label threats.
3. **Feature Engineering:** Extract features like transaction frequency, gas anomalies, contract vulnerabilities, network degree centrality.

4. **Model Training:** Train anomaly detection and classification models on historical blockchain attack datasets.

5. **Real-Time Deployment:** Integrate model with blockchain nodes for streaming analysis.

6. **Alert System:** Build pipeline to push alerts to dashboards and external systems.

7. **Evaluation:** Test detection speed, accuracy, and robustness against adversarial attacks.

Algorithmic Flow:

1. Data Acquisition (Blockchain logs, smart contracts, transactions).
2. Preprocessing (noise removal, normalization, categorical encoding).
3. Feature Extraction (network features, temporal patterns, anomaly scores).
4. Train ML/DL Models (Isolation Forest, Random Forest, LSTM).
5. Real-Time Threat Detection (streaming transactions).
6. Alert Generation (severity classification + real-time alerts).
7. Visualization (dashboard, attack reports).

Program:

```
!!pip install scikit-learn tensorflow matplotlib --quiet


import numpy as np

import pandas as pd

from sklearn.ensemble import IsolationForest, RandomForestClassifier

from sklearn.metrics import classification_report

from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import LSTM, Dense

# Synthetic blockchain dataset
np.random.seed(42)
n_samples = 1000

data = pd.DataFrame({
    "tx_volume": np.random.poisson(10, n_samples),
    "gas_used": np.random.normal(20000, 5000, n_samples),
    "contract_calls": np.random.poisson(3, n_samples),
    "network_latency": np.random.normal(50, 10, n_samples),
    "anomaly_score": np.random.uniform(0, 1, n_samples),
})

labels = np.random.choice([0,1], size=n_samples, p=[0.9,0.1])  # 0=Normal, 1=Threat

# Train Isolation Forest for anomaly detection
iso = IsolationForest(contamination=0.1, random_state=42)
iso.fit(data)
preds = iso.predict(data)

# Convert predictions (-1=anomaly, 1=normal) to binary
preds = [1 if p == -1 else 0 for p in preds]
```

print(classification_report(labels, preds))

Output:

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       900
           1       0.06      0.06      0.06       100

    accuracy                           0.81      1000
   macro avg       0.48      0.48      0.48      1000
weighted avg       0.81      0.81      0.81      1000
```