Anomaly Detection in System Logs

Name: Bhuvan Shyam M

Email: bhuvanshyam2022@gmail.com

Repository: https://github.com/BhuvanShyam/anomaly-detection-logs.git

1. Introduction

In large-scale IT infrastructure, system logs are critical digital footprints that document every action, event, or error occurring within hardware, applications, services, and networks. These logs provide valuable information regarding system behavior, user activity, performance metrics, failures, security events, and other operational insights.

With the advent of cloud computing, distributed systems, and microservices, modern infrastructures generate massive volumes of log data in real-time. These logs are often unstructured, heterogeneous, and noisy, making manual inspection highly impractical and inefficient. In traditional environments, rule-based log monitoring systems are used; however, they lack adaptability and cannot effectively capture novel patterns or zero-day anomalies.

Why This Matters:

- **Security**: System logs often contain evidence of unauthorized access, brute-force login attempts, or malware execution. Detecting these anomalies early can prevent larger breaches.
- **Reliability**: Unexpected system crashes or performance degradation events are frequently preceded by unusual log patterns.
- **Efficiency**: Automation reduces the burden on IT teams and helps them focus on incident resolution rather than detection.

2. Problem Statement

- Existing rule-based log monitoring solutions are limited because they cannot adapt to changes in system behavior or evolving security threats.
- The goal is to design an AI-based anomaly detection framework that:
- Learns normal log behavior.
- Detects deviations or unusual patterns.
- Handles diverse and unstructured log formats.

3. Objectives

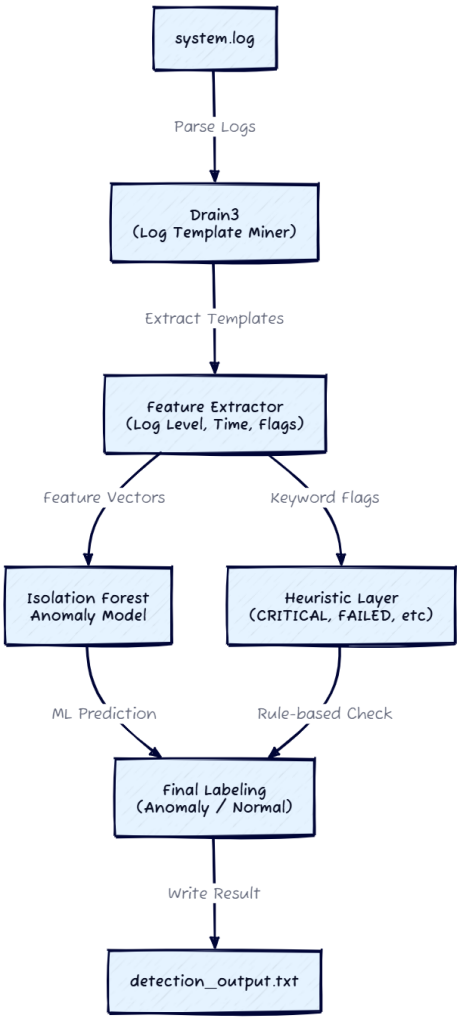
- Automatically detect anomalies in system logs.
- Enable log preprocessing and template mining.
- Minimize false positives by combining heuristics and ML.

- Output human-readable and explainable alerts.

4. Technologies Used

- Python: Programming language
- Drain3: Log template mining
- scikit-learn: Machine learning
- Isolation Forest: Anomaly detection
- Pandas: Data processing
- Regex: Pattern matching

5. System Architecture



System Architecture: Step-by-Step Breakdown

1. **Log Source (System Logs)**
    - These are raw log files generated by servers, applications, or devices.
    - Logs are unstructured and in plain text.
2. **Log Ingestion**
    - Logs are read from a file system or streamed into the pipeline.
    - Example: system.log is read using Python's file operations.
3. **Log Parser (Drain3 Template Miner)**
    - Converts raw log lines into structured templates.
    - Helps in identifying patterns and reducing dimensionality.
4. **Feature Extraction**
    - Extracts features like timestamp, log level, hour, day, and keyword flags (failed, crash, unauthorized).
    - Encodes templates and log levels for ML processing.
5. **Anomaly Detection Model (Isolation Forest)**
    - Trained to detect outliers in the log pattern features.
    - Outputs a prediction (Normal or Anomaly).
6. **Heuristic Post-Processing**
    - Adds rules to further label anomalies based on severity (e.g., CRITICAL logs are always anomalies).
7. **Output**
    - Logs are labeled with Normal or Anomaly.
    - Output is written to a text or CSV file for analysis.

6. Dataset and Preprocessing

- A synthetic dataset of 1000 log entries was generated.
- Logs parsed using Drain3 to extract templates.
- Features: log level, hour, day, crash/failed/unauthorized flags.
- Categorical values converted to numeric for ML.

7. Machine Learning Model

- Used Isolation Forest (unsupervised ML).
- Trained to identify outliers using extracted features.
- Prediction: 'Normal' or 'Anomaly'.
- Overlay heuristics: CRITICAL logs or failed login = anomaly.

8. Results and Output

- Output saved to 'detection_output.txt'.
- Each line tagged as [Anomaly] or [Normal].
- Detected anomalies: Failed logins, Unauthorized access, Crashes.

9. Repository Link

- GitHub: https://github.com/BhuvanShyam/anomaly-detection-logs.git

10. Conclusion and Future Work

- The system identifies unusual log patterns using template mining and ML.
- Future improvements:
- Real-time log monitoring
- Visualization dashboard
- Alert integration (Slack/email)
- Advanced models (autoencoders/transformers)

11. References

[1] Drain3 GitHub: https://github.com/logpai/drain3

[2] Isolation Forest Paper

[3] Scikit-learn: https://scikit-learn.org

[4] Python Docs: https://docs.python.org

[5] Synthetic logs generated