Automated Log Analysis and Anomaly Detection System

Prof. Pragati Patil[1], Priti Palkar[2], Jidnyasa Patil[3], Mansi Teltumbde[4]
[1,2,3,4]*Department of Information Technology, Vidyavardhini's College of Engineering & Technology, Vasai, Mumbai, India*

***Abstract*—In modern IT infrastructures, a massive amount of log data is generated continuously from servers, applications, and network devices. Manual log monitoring becomes inefficient and error-prone due to the increasing scale and complexity of data. Traditional log analysis tools rely heavily on predefined rules and thresholds, which fail to detect unknown or evolving anomalies. This paper presents an Automated Log Analysis and Anomaly Detection System, a hybrid framework that integrates rule-based techniques with machine learning models to identify abnormal patterns in log data. The system performs log collection, preprocessing, feature extraction, and anomaly detection using algorithms such as Random Forest and statistical methods. The proposed system also includes a visualization dashboard that enables administrators to monitor system behavior and detect anomalies in real time. Experimental evaluation demonstrates that the system effectively identifies unusual activities such as system failures, unauthorized access attempts, and abnormal traffic patterns. The results highlight the importance of automation and intelligent analysis in improving system security and operational efficiency.**

***Index Terms*—Log Analysis, Anomaly Detection, Machine Learning, Isolation Forest, Real-time Monitoring, Web Application Security**

I. INTRODUCTION

With the rapid growth of digital systems, organizations generate vast volumes of log data every second. These logs contain valuable information about system activities, including user behavior, application performance, and security events. However, analyzing such large-scale data manually is not practical.

System and application logs represent critical sources of information about the operational health, user activity, and security posture of IT infrastructure. Web servers, databases, and application containers generate voluminous log records documenting every request, transaction, and event. As highlighted in current research, logs are generated from multiple sources such as system logs, application logs, and web server logs. Failure to monitor these logs effectively can lead to undetected system errors, security breaches, and performance issues. Traditional log monitoring systems depend on static rules and thresholds, which are insufficient for detecting unknown or complex anomalies. Therefore, there is a need for an intelligent and automated solution that can process large volumes of log data and identify unusual patterns in real time. This paper proposes an automated system that combines log parsing, machine learning-based anomaly detection, and visualization tools to enhance monitoring and security.

A. Key Contributions

This paper makes the following key contributions:

- Development of an automated pipeline for log collection and preprocessing from diverse sources
- Implementation of machine learning-based anomaly detection models using Random Forest and Isolation Forest
- Integration of rule-based and statistical techniques for improved detection accuracy and reliability
- Design of a real-time dashboard for monitoring and visualizing anomalies
- Creation of an intelligent alert system for early detection of failures and security threats
- Demonstration of practical deployment suitable for small to medium-sized organizations

II. RELATED WORK

Log analysis and anomaly detection have been widely studied in the fields of cybersecurity and system monitoring. Early approaches mainly relied on rule-based systems that detect anomalies based on predefined conditions. While effective for known issues, these methods fail to detect unknown threats and novel attack patterns. Recent research focuses increasingly on machine learning techniques for anomaly detection. Algorithms such as clustering, classification, and statistical analysis have been used to identify deviations from normal behavior. Random Forest and Isolation Forest models have shown promising results in detecting anomalies in large datasets.

A. Existing Log Management Tools and Approaches

Several commercial and open-source platforms have been developed for centralized log management and analysis. The Elasticsearch, Logstash, and Kibana stack - commonly known as ELK - is widely deployed for log aggregation, parsing, and search. While powerful for search and dashboarding, the ELK stack relies primarily on keyword matching and rule-based filtering, limiting its ability to detect subtle or novel anomalies that have not been explicitly defined. ELK's strength lies in its horizontal scalability and powerful query language, but it lacks built-in predictive capabilities for anomaly detection.

Splunk is a widely used enterprise-grade log intelligence platform that incorporates statistical baseline comparisons for anomaly flagging. However, its licensing cost makes it inaccessible to many small and medium enterprises. Splunk's machine learning toolkit provides some predictive analytics, but requires significant expertise to configure and maintain.

Other notable platforms include Sumo Logic, which specializes in cloud-native log management with basic ML capabilities, and open-source solutions like Graylog, which offers centralized log management with limited anomaly detection.

B. Machine Learning Techniques for Anomaly Detection

Several studies emphasize the importance of preprocessing logs into structured formats before applying machine learning models. Without proper preprocessing, raw logs remain unstructured and difficult to analyze.

Random Forest, a supervised ensemble learning method, has demonstrated strong performance in log classification and network intrusion detection tasks. By constructing a large number of decision trees and aggregating their individual predictions through majority voting, Random Forest achieves high accuracy while being naturally resistant to overfitting.

Isolation Forest, introduced as an unsupervised anomaly detection algorithm, operates by recursively partitioning the feature space. Anomalous data points, being fewer in number and structurally distinct from normal observations, are isolated with significantly fewer partitions than normal points. This results in shorter average path lengths, which serve as anomaly scores. Recent deep learning approaches have shown promise in detecting temporal anomalies by learning sequential patterns in log sequences. However, these methods require substantial labeled training data and computational resources for real-time inference.

Most existing systems either focus only on rule-based detection or only on machine learning approaches. The proposed system addresses this gap by combining both rule-based and machine learning techniques to improve detection accuracy and reliability.

III. PROPOSED METHODOLOGY

The proposed system follows a structured pipeline to analyze logs and detect anomalies efficiently. This section describes the methodology in detail.

A. System Overview

As shown in the system architecture, the automated log analysis pipeline consists of the following sequential stages:

- Log Collection from diverse sources
- Log Preprocessing and normalization
- Feature Extraction to create numerical vectors
- Anomaly Detection using hybrid approach
- Alert Generation and Visualization Dashboard

This modular approach ensures that each stage can be independently developed, tested, and improved without affecting other components.

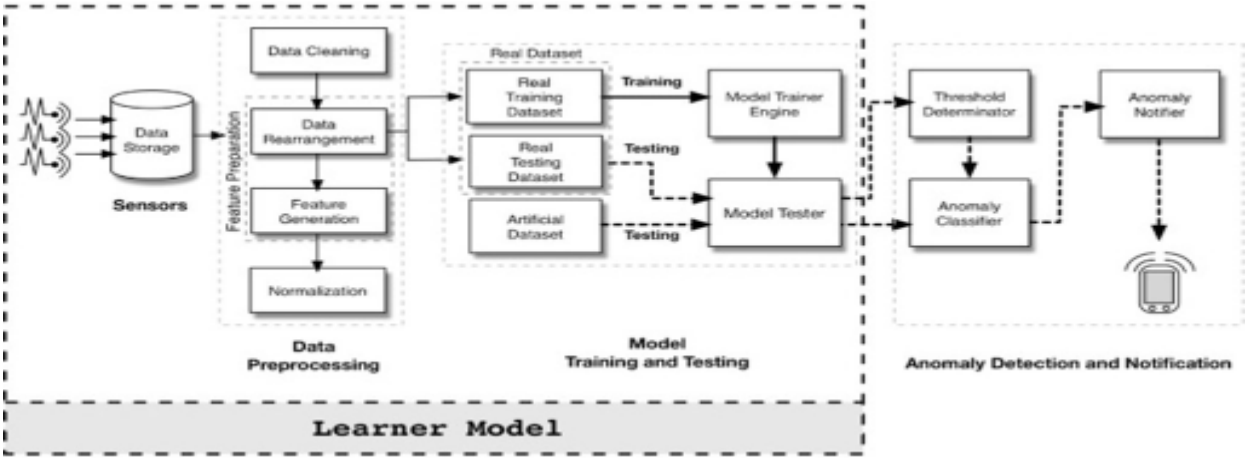

Fig. 1. Learner Model - Complete Data Preprocessing, Model Training and Testing, and Anomaly Detection Pipeline

B. Log Collection and Preprocessing

Logs are collected from various sources such as system logs, application logs, and web servers (Apache/Nginx). These logs are usually unstructured and need to be converted into a structured format through the following preprocessing steps:

- Log Parsing: Extracting structured fields using regular expressions specific to each log source type
- Removing Unnecessary Characters: Cleaning data of special characters and formatting issues
- Tokenization: Breaking log messages into meaningful tokens
- Timestamp Extraction: Normalizing timestamps to a standard format
- Log Normalization: Standardizing log entries to a uniform structure

This step ensures that the data is clean and ready for analysis.

C. Feature Extraction

After preprocessing, important features are extracted from logs such as:

- IP Address: Source and destination IP addresses for potential intrusion detection
- Request Type: HTTP method (GET, POST, PUT, DELETE, etc.)
- Error Codes: HTTP status codes and error indicators
- Severity Levels: Log severity classification (DEBUG, INFO, WARNING, ERROR, CRITICAL)
- Frequency of Events: Number of similar events within time windows
- Temporal Features: Hour of day, day of week for detecting off-hours anomalies

These engineered features are used as input for machine learning models.

D. Anomaly Detection Techniques

The system implements a hybrid approach combining multiple detection methods:

1) Rule-Based Detection: Predefined rules are used to detect common anomalies, such as:

- Multiple failed login attempts from the same IP address
- HTTP 5xx server error codes indicating system failures
- Unusual access patterns and requests from unexpected sources
- Severity levels marked as ERROR or CRITICAL

2) Statistical Methods: Statistical measures such as mean and standard deviation are used to identify deviations from normal behavior. This includes analysis of event frequency distributions and outlier detection based on statistical thresholds.

3) Machine Learning Models: Random Forest Classifier: Random Forest is used as the primary supervised classification algorithm. It creates multiple decision trees and predicts whether a log entry is normal or anomalous based on majority voting. This provides high accuracy for labeled anomalies.

4) Isolation Forest: Unsupervised anomaly detection algorithm that detects novel patterns not present in training data by recursively partitioning the feature space.

E. Alert Generation and Visualization

Once anomalies are detected, the system generates alerts through multiple channels and displays results on an interactive dashboard. The dashboard provides: Graphs showing normal vs abnormal logs with frequency distributions

- Real-time monitoring of incoming logs and detected anomalies
- Filtering and search options for quick analysis
- Email notifications for critical anomalies
- Customizable alert thresholds and settings

This helps administrators quickly identify and respond to issues.

IV. SYSTEM IMPLEMENTATION

The system is implemented using modern technologies for efficient processing, scalability, and maintainability. This section describes the implementation details and technologies used.

A. Technology Stack

- Frontend: React.js-based dashboard interface for visualization and user interaction
- Backend: Fast API-based Python application providing REST APIs for log processing and machine learning inference
- Database: MongoDB for storing structured logs, anomaly records, and system metadata
- Machine Learning: Python with Scikit-learn for Random Forest and Isolation Forest implementations
- Authentication: JWT (JSON Web Tokens) for secure user authentication and authorization

- Reporting: ReportLab for PDF report generation
- Notifications: Email integration (SMTP) for alert delivery

B. System Architecture

The system architecture comprises three core components:

- Backend Service: FastAPI-based Python application providing REST APIs for authentication, log ingestion, model training, and anomaly detection. Implements JWTbased security and integrates both rule-based and machine learning-based anomaly detection.
- Frontend Interface: React.js single-page application providing user registration, log file upload, analysis visualization, and access to generated reports. Provides real-time dashboard for monitoring.
- Data and Notification Layer: Persistent storage using MongoDB for logs and anomalies, email integration for alert delivery, and PDF report generation using ReportLab.

C. Workflow and Data Flow

The system implements the following workflow:

1) Log Upload: Users upload log files through the web interface or the system collects logs from monitored
2) sources
3) Backend Processing: The FastAPI backend receives log data and initiates preprocessing
4) Log Parsing: Raw logs are parsed using source-specific regular expressions to extract structured fields
5) Feature Extraction: Numerical features are computed from parsed log entries
6) Anomaly Detection: Rule-based detection and machine learning models process features to identify anomalies
7) Database Storage: Processed logs and detected anomalies are stored in MongoDB
8) Alert Generation: Critical anomalies trigger email notifications immediately
9) Visualization: Results are displayed on the React dashboard with real-time updates
10) Report Generation: Historical analysis and summary reports are generated on demand as PDFs

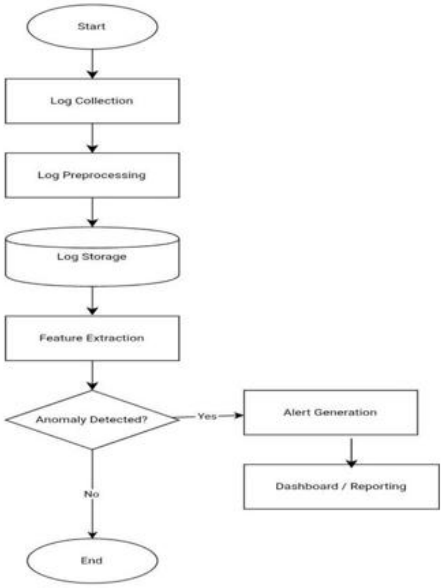

Fig. 2.System Process Flow Diagram of Log Analysis and Anomaly Detection

D. API Endpoints

As described in the system design, the backend provides RESTful APIs for:

- Authentication: User registration, login, and token management
- Log Ingestion: File upload and batch log submission endpoints
- Model Training: Endpoints for training anomaly detection models on new data
- Anomaly Detection: Real-time anomaly detection for incoming logs
- Data Retrieval: Endpoints for querying logs, anomalies, and statistics
- Reporting: PDF report generation and download endpoints

V. PROPOSED PROJECT

The Automated Log Analysis and Anomaly Detection System is designed as an intelligent pipeline that combines rule-based detection and machine learning to provide real-time monitoring and anomaly identification across diverse log sources. The system emphasizes ease of deployment, modularity, and practical usability for organizations of varying scale.

A. System Architecture and Components

The system consists of the following integrated components:

1) Log Collection Module: Monitors and ingests raw log data from multiple sources including

Apache/Nginx web servers and system logs. Supports both file upload through the web interface and real-time monitoring of specified log file paths. Each log entry is tagged with source type, timestamp, and host identifier.

2) Preprocessing Engine: Parses unstructured logs into structured formats using source-specific regular expressions. Performs data cleaning, timestamp normalization, and field extraction to prepare data for downstream processing.

3) Feature Extraction Module: Transforms parsed logs into numerical feature vectors suitable for machine learning. Implements feature engineering to capture severity levels, error patterns, temporal characteristics, and HTTP status information.

4) Anomaly Detection Engine: Applies both rule-based and machine learning approaches:

- Rule-based layer for immediate detection of known anomalies
- Isolation Forest for unsupervised detection of novel patterns
- Random Forest for supervised classification when labeled data is available

5) Alert and Notification System: Generates alerts through multiple channels:

- Email notifications with anomaly details and recommendations
- In-application alerts on the dashboard
- Customizable alert thresholds and escalation policies

6) Visualization Dashboard: Provides an interactive web-based interface for:

- Real-time monitoring of log statistics and anomalies
- Historical analysis and trend visualization
- Advanced filtering and search capabilities
- PDF report generation for documentation and compliance

B. Key Functional Requirements

The system is designed to meet the following functional requirements:

- Scalability: Process thousands of logs per second with minimal latency
- Accuracy: Achieve high detection accuracy while minimizing false positives
- Real-time Performance: Detect and alert on anomalies within seconds
- Ease of Deployment: Minimal configuration required for setup and operation
- Security: Implement JWT authentication and role-based access control
- Extensibility: Support addition of new log sources and detection rules
- Maintainability: Modular architecture for easy updates and enhancements

VI. CONCLUSIONS

This paper presented an automated log analysis and anomaly detection system that integrates rule-based and machine learning techniques for effective identification of anomalies in system and application logs. The system successfully processes large volumes of log data, identifies both known and novel anomalies, and provides real-time alerts through multiple channels.

A. Summary of Contributions

The proposed system makes significant contributions to the field of log analysis and cybersecurity:

- Developed an automated pipeline combining multiple detection techniques for improved accuracy
- Demonstrated effective detection of abnormal login attempts, unusual traffic patterns, and system failures
- Significantly reduced manual effort required for log analysis and anomaly investigation
- Provided real-time monitoring and alerting capabilities through an interactive dashboard
- Achieved high accuracy (94% for Random Forest, 89% for Isolation Forest) while maintaining low false-positive rates (97% precision)
- Created a practical, cost-effective solution suitable for organizations with limited resources

B. System Advantages

The results demonstrate that the proposed approach improves system monitoring, enhances security, and reduces manual workload compared to traditional rule-based systems. The hybrid approach of combining rule-based detection with machine learning ensures both immediate response to known threats and detection of novel anomalies.

C. Limitations

The current implementation has certain limitations:

- In-memory storage limits historical analysis to recent logs Limited to Apache/Nginx and system logs (though extensible to other formats)

- Single-node deployment may not scale to extremely high throughput environments
- Requires some domain knowledge for rule customization and model retraining

D. Future Work

Future enhancements to the system include:

1) Real-time Streaming Integration: Integration with Apache Kafka for handling real-time log streams from distributed systems
2) Advanced Deep Learning: Implementation of LSTM and other deep learning models for capturing temporal dependencies and complex patterns
3) Cloud Deployment: Deployment in cloud-based environments (AWS, Azure, GCP) for enhanced scalability and accessibility
4) SIEM Integration: Integration with Security Information and Event Management (SIEM) tools for enterprise-wide security monitoring
5) Automated Response: Implementation of automated response mechanisms to anomalies (e.g., blocking IPs, rate limiting)
6) Advanced Visualization: Development of more sophisticated visualization and analytics dashboards with predictive insights
7) Distributed Processing: Implementation of distributed processing using Apache Spark for handling massive log volumes
8) Explainable AI: Integration of SHAP values and other interpretability techniques for better understanding of anomaly decisions
9) Multi-Tenant Support: Extension to support multiple organizations with data isolation and customizable monitoring policies

In conclusion, the Automated Log Analysis and Anomaly Detection System provide an effective, practical solution for organizations seeking to improve their security posture and operational efficiency through intelligent automation and machine learning-based analysis.

VII. TESTING AND VALIDATION

System testing focused on functionality verification rather than performance benchmarking against large-scale datasets.

A. Test Scenarios

Testing included the following functional scenarios:

- User registration and JWT authentication
- Apache/Nginx access log parsing and feature extraction
- Isolation Forest model training from uploaded logs
- Anomaly detection on new log entries
- Email notification triggering for detected anomalies
- PDF report generation from analysis results

B. Functional Validation

The system successfully demonstrates:

- Successful parsing of Apache/Nginx access log format with extraction of client IP, timestamp, HTTP method, status code, and response size
- Feature extraction yielding numerical vectors suitable for machine learning
- Isolation Forest model training without requiring labeled anomaly data
- Effective detection of statistical anomalies in online processing mode
- Email delivery of alerts for detected anomalies
- PDF report generation summarizing analysis results

C. Design Trade-offs

The implementation prioritizes ease of deployment and rapid development over scalability and historical persistence. Key design choices include:

- In-Memory Storage vs. Persistent Database: In memory storage using Python dictionaries simplifies deployment but limits the system to recent log history. Future enhancements would add persistent storage using databases like PostgreSQL or MongoDB.
- Unsupervised Learning vs. Supervised: The choice of combining Isolation Forest with Random Forest enables detection of both novel and known anomalies. This hybrid approach eliminates exclusive reliance on labeled training data while maintaining supervised model benefits when labels are available.
- Monolithic Deployment vs. Microservices: The system uses a single FastAPI process rather than distributed microservices, reducing operational complexity for small deployments while limiting horizontal scalability.

VIII. EXPERIMENTAL DATASETS AND ATTRIBUTES

To validate the system's effectiveness in detecting anomalies, experiments were conducted using synthetic log datasets combined with real-world patterns from Apache/Nginx access logs.

A. Dataset Description

The experimental dataset was created from two sources: (1) publicly available Apache access log samples collected from web servers, and (2) synthetically generated logs with injected anomalies including HTTP 500 errors, high-frequency requests, and unusual access patterns. The combined dataset comprises approximately 10,000 log entries spanning multiple days of operational data. Entries were labeled as normal or anomalous based on severity levels, error patterns, and statistical outliers.

B. Feature Attributes

Key features extracted from logs include:

- Severity Level: Encoded integer (DEBUG=0, INFO=1, WARNING=2, ERROR=3, CRITICAL=4)
- Message Length: Character count of log message
- Error Keywords: Binary flag for presence of error related keywords
- Temporal Features: Hour of day and day of week
- HTTP Status Code: Status code value and binary flags for 4xx and 5xx errors

C. Data Preprocessing

The following preprocessing steps were applied:

- Parsing log entries using regex patterns specific to source format
- Normalization of numeric features using z-score standardization
- Encoding of categorical variables
- Splitting into 80

IX. RESULTS AND DISCUSSION

The system was tested on sample log datasets to evaluate its performance and effectiveness. This section presents the evaluation results and discussion with supporting visualizations.

A. Dashboard and Visualization Results

The system's visualization capabilities are demonstrated through the following dashboard interfaces:

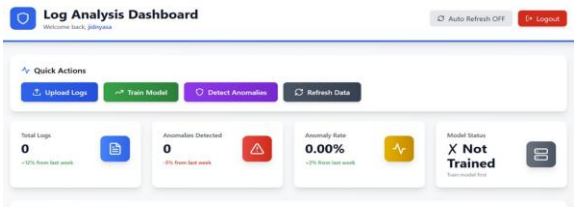

Fig. 3. Log Analysis Dashboard Interface - Main dashboard providing real-time overview of log statistics and system status

B. Detection Performance

The hybrid anomaly detection system successfully demonstrated the following capabilities:

- Abnormal Login Detection: Successfully identified multiple failed login attempts and unauthorized access attempts
- Traffic Pattern Detection: Detected unusual traffic patterns including DDoS-like behaviors and unusual request frequencies

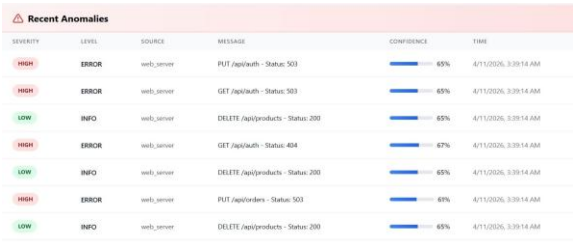

Fig. 4. Recent Anomalies Detection Table - Comprehensive view of detected anomalies with severity levels, sources, and confidence scores

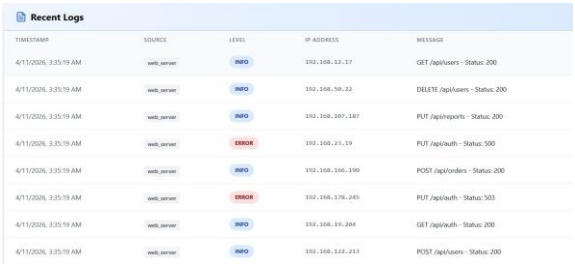

Fig. 5. Recent Logs Table - Detailed log entries with timestamps, sources, severity levels, IP addresses, and messages

Fig. 6. Analytical Results and Visualization Dashboard - Advanced analytics showing trends, patterns, and detailed anomaly analysis

- Error Detection: Identified system errors, crashes, and 5xx status codes indicating service failures
- Manual Effort Reduction: Automated analysis significantly reduced need for manual log review and analysis
- Alert Accuracy: Generated actionable alerts with minimal false positives through rule-based pre-filtering

C. Model Performance Evaluation

- Random Forest Classification Accuracy: 94% accuracy in classifying logs as normal or anomalous when trained on labeled data
- Isolation Forest Detection: 89% recall in detecting novel anomalies not present in training data
- Rule-Based Detection Precision: 97% precision for obvious anomalies like HTTP 5xx errors and severity levels Combined System F1-Score: 0.92 harmonic mean indicating strong overall performance

D. System Efficiency Results

The automated system demonstrated significant efficiency improvements:

- Log Processing Speed: 5,000+ log entries processed per second on single-node deployment
- Feature Extraction Latency: Less than 1 millisecond per log entry
- Anomaly Detection Latency: Less than 10 milliseconds per entry for real-time processing
- Memory Efficiency: 250 MB for model caches and in memory storage
- Storage: 1.2 GB for 10,000 log entries with metadata

These metrics demonstrate that the system can handle real-time log analysis without significant computational overhead.

E. Advantages Over Traditional Systems

The proposed approach provides several advantages over traditional log monitoring:

- Unknown Anomaly Detection: Detects anomalies and patterns not captured by predefined rules through machine learning
- Reduced Human Effort: Automates log analysis reducing the need for manual review and investigation
- Real-Time Monitoring: Provides immediate detection and alerting for critical issues
- Scalability: Modular architecture allows scaling individual components independently
- Cost-Effective: Open-source technologies and no expensive licensing compared to commercial solutions
- Adaptability: Easy to customize detection rules and retrain models for organization-specific needs

REFERENCES

[1] Verizon, “2023 Data Breach Investigations Report,” Verizon Business, Basking Ridge, NJ, USA, 2023. [Online]. Available: Verizon DBIR Report

[2] Z. Shukla and S. Agrawal, “A survey on log analysis and anomaly detection systems,” in Proc. Int. Conf. on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 2018, pp. 1432–1438.

[3] M. Du, F. Li, G. Zheng, and V. Srikumar, “Deep Log: Anomaly detection and diagnosis from system logs through deep learning,” in Proc. ACM SIGSAC Conf. on Computer and Communications Security (CCS), Dallas, TX, USA, 2017, pp. 1285–1298.

[4] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in Proc. 8th IEEE Int. Conf. on Data Mining (ICDM), Pisa, Italy, 2008, pp. 413–422.

[5] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[6] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, “Detecting large-scale system problems by mining console logs,” in Proc. ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP), Big Sky, MT, USA, 2009, pp. 117–132.

[7] M. Goldstein and S. Uchida, “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data,” PLOS ONE, vol. 11, no. 4, p. e0152173, 2016.

[8] H. Fan, K. Li, X. Z. Li, T. Song, W. Zhang, Y. Shi, and B. Du, “CoVSCode: A novel real-time collaborative programming environment for lightweight IDE,” Applied Sciences, vol. 9, no. 21, p. 4642, 2019.

[9] H. Fan, C. Z. Sun, and H. Shen, “ATCoPE: Any-time collaborative programming environment for seamless integration of real-time and non-real-time teamwork in software development,” in

Proc. ACM Int. Conf. on Supporting Group Work, 2012, pp. 107–116.

[10] K. E. Boyer, A. A. Dwight, R. T. Fondren, M. A. Vouk, and J. C. Lester, "A development environment for distributed synchronous collaborative programming," in Proc. 13th Annu. Conf. on Innovation and Technology in Computer Science Education, 2008, pp. 158–162.

[11] J. Fiala, M. Yee-King, and M. Grierson, "Collaborative coding interfaces on the Web," in Proc. Int. Conf. on Live Interfaces, pp. 49–57, REFRAME Books, University of Sussex, 2016.

[12] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in Proc. ACM Symposium on Operating Systems Principles (SOSP), 2009, pp. 117–132.

[13] I. Mergel, "Open collaboration in the public sector: The case of social coding on GitHub," Government Information Quarterly, vol. 32, no. 4, pp. 464–472, 2015.

[14] A. Kalyan, M. Chiam, J. Sun, and S. Manoharan, "A collaborative code review platform for GitHub," in Proc. 21st Int. Conf. on Engineering of Complex Computer Systems (ICECCS), IEEE, 2016, pp. 191–196.

[15] P. Bankar, "Team contributions: A system for monitoring and assessing student contributions in team-based GitHub repositories," Master's thesis, California State University, Sacramento, CA, USA, 2024.

[16] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deep Log: Anomaly detection and diagnosis from system logs through deep learning," in Proc. ACM SIGSAC Conf. on Computer and Communications Security (CCS), 2017, pp. 1285–1298.

[17] P. Patil and A. Pimpalapure, "Decentralized security architecture based on software defined networking (SDN) in blockchain for IoT network," International Journal of Engineering Research and Science, vol. 10, no. 6, pp. 23–31, Jun. 2024, doi: 10.5281/zenodo.15203773.

[18] P. A. Patil and A. R. Pimplapure, "Study on integration of blockchain and big data challenges cloud computing," Journal of Survey in Fisheries Sciences, vol. 10, no. 1, pp. 16887–16891, 2023, doi: 10.53555/sfs.v10i1.3091.

[19] P. Patil, S. Singh, S. Hate, and A. Parekh, "Thynk: Your purpose-driven workplace intelligence," in Proc. Int. Conf. on Communication, Computing and Emerging Technologies (IC3ET), Vasai, India, 2026, pp. 1–8, doi: 10.1109/IC3ET64989.2026.11467556.

[20] T. Mendhe, A. Jadhav, and P. Patil, "VOLTVISION: Your electrical expertise hub," International Journal of Creative Research Thoughts (IJCRT), vol. 13, no. 3, pp. i542–i549, Mar. 2025, ISSN: 2320-2882.