El-Sewedy University of Technology

Faculty of Engineering Technology
Department of Computer Engineering Technology

Graduation Project II

Project Title:

Enhanced DevOps Security Using AI

Prepared By:

Ahmed Essam Eldin Mohamed	230103559
Dalia Ali Abdallah	230103981
Kamel Ahmed Mostafa	230100952
Michael Gameel Moussa	230105811
Mohamed Hany Refaat	230104513
Mohamed Hesham Abbas	230105474
Omar Mohamed Abdo	230103309
Seif Wahed Mahmoud	230102645
Yassine Shaher Mohy Eldeen	230103609

Supervised By:

Dr. Maryam Adel

Abstract

The rapid evolution of DevOps practices has significantly increased the complexity and speed of software delivery, making it increasingly difficult to maintain effective security measures within CI/CD pipelines. Traditional rule-based security systems struggle to scale and adapt to modern DevOps environments, leaving systems vulnerable to advanced threats.
This project proposes an AI-driven security solution that leverages we utilize three machine learning models—Isolation Forest, DeepLog, and Transformer networks—to detect anomalies in CI/CD logs. This approach provides a lightweight and scalable solution for identifying security threats such as unauthorized access and abnormal resource usage. In future phases, Long Short-Term Memory (LSTM) and enhanced Transformer architectures will be integrated to analyze multi-step attack patterns, enabling automated root-cause analysis and incident remediation.

The system aims to provide continuous monitoring and proactive threat detection, reducing the need for manual intervention and enhancing overall security posture. Additionally, feedback mechanisms will be implemented to ensure continuous improvement of the system in response to evolving threats. This layered, AI-driven approach ensures that security measures in DevOps environments remain adaptive, resilient, and aligned with best practices throughout the software development lifecycle.

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Table of Contents

List of figures

List of Abbreviations

Abbreviations	Definitions
DevOps	Development Operations
AI	Artificial Intelligence
CI/CD	Continuous Integration/Continuous Delivery
DDOS	Distributed Denial of Service
LSTM	Long Short-Term Memory
ML	Machine Learning
UAT	User Acceptance Testing.
OS	Operating System
API	Application Programming Interface
JSON	JavaScript Object Notation

Enhanced DevOps Security Using AI

Chapter 1: Introduction

Introduction

The AI Security initiative leverages Isolation Forest and is used as a lightweight, unsupervised algorithm to detect anomalies in CI/CD logs. Future phases will incorporate LSTM and Transformer networks to analyze multi-step attack patterns, enabling automated root-cause analysis and remediation. This layered approach enhances threat detection accuracy while minimizing the need for human intervention.
The initiative also focuses on integrating advanced security measures into the CI/CD pipeline, aiming to identify, mitigate, and respond to security risks effectively, ensuring adherence to security best practices throughout the software development lifecycle. Additionally, a continuous monitoring system will be developed using AI to analyze logs and network traffic for real-time threat detection, ensuring proactive security measures.

Background:

Traditional rule-based systems struggle to scale with the increasing complexity of modern DevOps environments. Isolation Forest efficiently identifies outliers in high-volume logs, such as unauthorized access or resource spikes, while neural networks address more contextual threats like credential stuffing or DDoS attacks. AI bridges the gap between detection and action, enabling autonomous incident resolution. As DevOps practices accelerate software delivery, securing the CI/CD pipeline becomes more challenging, necessitating robust and integrated security measures to address these evolving threats effectively. Feedback mechanisms will also be established to continuously improve security measures based on evolving threats, ensuring the system remains adaptive and resilient.

1.1 Problem Statement

Organizations face several critical challenges in managing and analyzing security logs:

- Log Management Complexity: Traditional log management systems struggle with the volume, velocity, and variety of modern log data. Manual analysis is time-consuming and error-prone, while centralized collection and storage present scalability challenges.
- Anomaly Detection Accuracy: Single-model anomaly detection approaches suffer from high false positive rates and limited detection capabilities. They often fail to identify sophisticated attack patterns that span multiple data dimensions or evolve over time.
- Integration Challenges: Existing solutions often operate in silos, making it difficult to correlate information across different data sources and detection methods.

1.2 Motivation and Justification

To enhance the effectiveness and relevance of the original project scope, several strategic modifications have been introduced. These changes are aimed at improving threat detection accuracy, scalability, and automation within modern DevOps environments.

1. Integration of Isolation Forest for Anomaly Detection

- Modification:
  Adopted the Isolation Forest algorithm as the initial lightweight model for detecting anomalies in CI/CD logs.
- Justification:
  Isolation Forest is an unsupervised and computationally efficient algorithm that excels in identifying outliers in large-scale log data, such as unusual resource usage or unauthorized access attempts. Its low overhead makes it suitable for real-time monitoring without disrupting CI/CD performance.

2. Integration of LSTM and Transformer Models

- Modification:
  Integrated deep learning models (LSTM and Transformer) to analyze sequential attack patterns, enabling automated root-cause analysis and response. This layered approach enhances threat detection accuracy while minimizing the need for human intervention.
- Justification:
  While Isolation Forest is effective for immediate anomaly detection, LSTM and Transformer models are better suited for understanding temporal dependencies and multi-step attack behaviors, such as lateral movement or credential stuffing. This layered AI approach provides a deeper level of insight and response capability.

3. CI/CD Pipeline Security Embedding

- Modification:
  Embedding AI-driven security checks directly into the CI/CD pipeline stages.
- Justification:
  Traditional post-deployment security checks are reactive and often too late. By integrating security into the CI/CD process, vulnerabilities can be identified and mitigated early, aligning with DevOps principles and improving software quality and compliance.

4. Development of Continuous Monitoring System

- Modification:
  Building a continuous AI-powered monitoring system for analyzing logs and network traffic.

- Justification:
  Continuous analysis is critical for rapid threat detection and response. AI-based

continuous monitoring enables proactive identification of threats, reducing the window of exposure and supporting zero-trust principles in DevOps.

1.2.1 Example of a subpoint

1. Integration of Isolation Forest for Anomaly Detection
   - Log sources integrated: GitLab CI/CD runner logs, Jenkins pipeline logs.
   - Features extracted: Execution time, CPU usage, user access patterns.
   - Anomalies detected: Unauthorized access, resource spikes, failed login attempts.
   - Toolchain used: Python, Logstash for pre-processing.

2. Future Integration of LSTM and Transformer Models
   - Data preparation: Sequence labeling of multi-step events (e.g., login → privilege escalation → data exfiltration).
   - Model training
   - Use case: Detection of advanced persistent threats (APT), lateral movement across systems.
   - Deployment plan: Containerized microservice that plugs into the pipeline monitoring system.

3. CI/CD Pipeline Security Embedding
   - Stages enhanced:
     - Build stage
     - Test stage
     - Deploy stage: Final anomaly detection checkpoint before push to production.
   - Automation tool
   - Alerts

4. Development of Continuous AI Monitoring System

   - Data sources:
     We work with logs collected from the web application to detect anomalies effectively.

   - AI engine: Combines Isolation Forest and Transformer for layered detection.
   - Response mechanism

5. Feedback-Driven Threat Adaptation
   - Feedback loop design: Analyst or admin marks a detection as false positive/true positive.
   - Model retraining schedule: Weekly or after major incidents.
   - Versioning: Use MLflow for model version tracking and performance comparison.
   - Continuous Learning: Incorporate feedback to fine-tune thresholds or retrain on edge-case scenarios.

1.2.2 Traditional Anomaly Detection Limitations

1.2.2.1 Single Model Approach Limitations

Single-model anomaly detection systems, while computationally efficient, suffer from several fundamental limitations:

- Limited Pattern Recognition: Each model type excels at detecting specific types of anomalies but may miss others.
- High False Positive Rates: Without correlation from multiple perspectives, single models often generate numerous false alarms.
- Lack of Confidence Metrics: Single models typically provide binary classifications without confidence levels.
- Vulnerability to Adversarial Attacks: Attackers can potentially evade detection by understanding and exploiting single model weaknesses.

1.3 Vision

To build an intelligent, adaptive, and fully integrated security framework for modern DevOps pipelines by leveraging artificial intelligence for real-time anomaly detection, proactive threat mitigation, and autonomous incident response.

This system will embed security into every stage of the CI/CD lifecycle, transforming traditional reactive measures into a dynamic, self-learning defense mechanism that evolves alongside emerging cyber threats ultimately enabling faster, safer software delivery with minimal human intervention.

1.4 Mission

To develop a lightweight, AI-driven security solution that integrates seamlessly into CI/CD pipelines, using models like Isolation Forest, LSTM, and Transformers to detect anomalies, identify multi-step attacks, and automate root-cause analysis.

The mission is to enhance the security posture of DevOps environments through continuous monitoring, intelligent feedback loops, and autonomous threat response ensuring secure, efficient, and resilient software delivery.

1.5 SWOT Analysis

Strengths:

- Proven ELK Stack technology with strong community support and extensive documentation.
- Multi-model approach providing comprehensive anomaly detection coverage.
- Open-source foundation reducing licensing costs and enabling customization.
- Scalable architecture supporting growth in data volume and infrastructure complexity.

- Processing capabilities enable prompt threat response.

Weaknesses:

- High resource requirements for optimal performance, particularly memory and storage.
- Complex configuration and tuning requirements demanding specialized expertise.
- Initial setup complexity may require significant time investment.
- Machine learning models require training data and ongoing refinement.

Opportunities:

- Growing market demand for advanced security analytics solutions.
- Integration possibilities with existing security information and event management (SIEM) systems.
- Potential for extending the system with additional data sources and detection models.
- Cloud deployment options enabling software-as-a-service offerings.

Threats:

- Rapidly evolving threat landscape requiring continuous model updates.
- Competition from commercial solutions with extensive support organizations.
- Potential performance degradation under extreme load conditions.
- Security vulnerabilities in open-source components requiring ongoing monitoring

1.6 Project Outcome

The successful completion of this project delivers a comprehensive logging and anomaly detection solution with the following key outcomes:

Technical Achievements:

- Fully operational ELK Stack deployment with optimized configurations for performance and security.
- Implementation of three distinct machine learning models (DeepLog, Transformer, Isolation Forest) for anomaly detection.
- Advanced correlation engine providing risk-based anomaly assessment with confidence scoring.
- Log processing pipeline capable of handling high-volume data streams..

Performance Metrics:

- Improved anomaly detection accuracy compared to single-model approaches.
- Reduced false positive rates through multi-model correlation.
- Processing capabilities with sub-second response times.
- Scalable architecture supporting thousands of log sources.

Business Value:

- Enhanced security posture through proactive threat detection.
- Reduced incident response times enabling faster threat containment.
- Comprehensive visibility into system activities supporting compliance requirements.
- Cost-effective solution leveraging open-source technologies.

1.7 Document Organization

This document is organized into six main chapters and appendices:

- Chapter 1 provides an introduction to the project, including problem definition, motivation, vision, mission, and SWOT analysis.
- Chapter 2 presents a comprehensive feasibility study covering technical, economic, and operational considerations.
- Chapter 3 details the selection process for hardware, software, and machine learning technologies used in the implementation.
- Chapter 4 describes the methodology and system design, including block diagrams, flow charts, and architectural decisions.
- Chapter 5 covers the complete system implementation, including detailed configuration steps, code implementation, and integration procedures.
- Chapter 6 concludes the document with project outcomes, lessons learned, and recommendations for future enhancements.

Chapter2: Feasibility Study

2.1 Executive Summary

This project focuses on enhancing the security of DevOps environments by integrating Artificial Intelligence (AI) and Machine Learning (ML). The primary goal is to automate tasks like threat detection, vulnerability scanning, and incident response, creating a smoother and more secure software development process. By leveraging AI/ML, the project aims to detect unusual activities, monitor logs and network traffic in real time, and automate security checks within CI/CD pipelines.

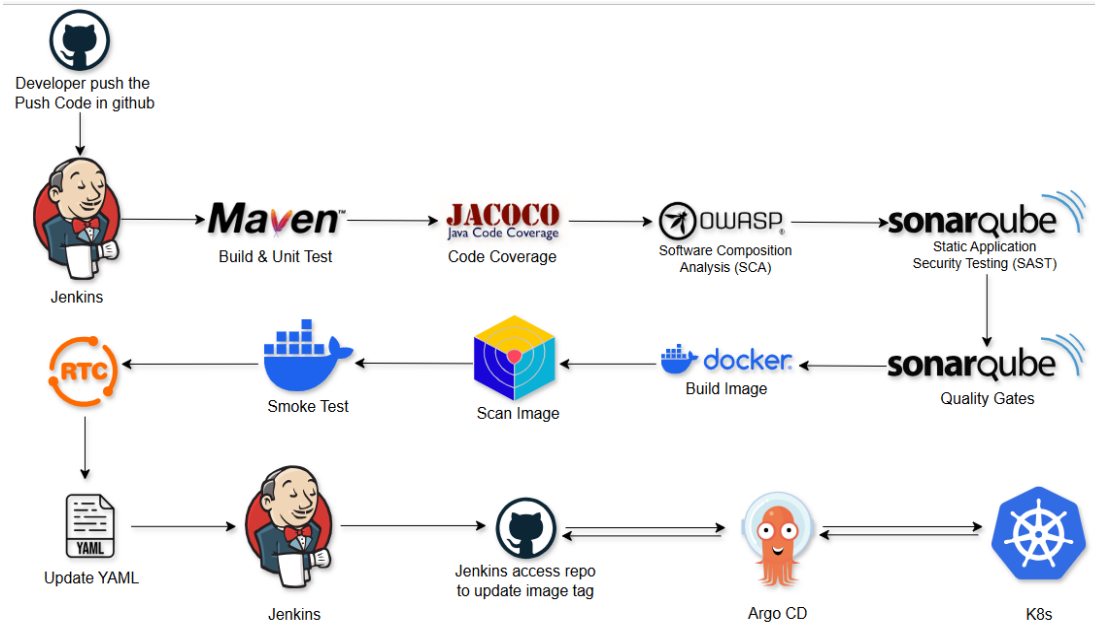
To achieve these goals, the project utilizes tools such as Python for developing AI/ML models, Jenkins and GitLab CI/CD for automation, and security tools like SonarQube, OWASP ZAP, and Snort. Additionally, the ELK Stack is employed for log management and real-time monitoring. The focus also includes training teams and establishing feedback loops to continually improve security practices, fostering collaboration between DevOps and security teams.

By integrating AI/ML into DevOps, the project sets a new standard for secure, efficient, and collaborative software development. It aims to achieve faster threat responses, reduce manual effort, and enhance team coordination. As organizations increasingly adopt cloud-native and container-based systems, the adaptive security measures introduced in this project will address emerging challenges and regulatory requirements, paving the way for a future where software delivery is both fast and resilient.

2.2 Description of tools

Table 1 Description of Tools

Technology	Purpose
Jenkins	CI/CD automation
SonarQube	Code quality and security analysis
Kubernetes	Container orchestration
ELK Stack	Log management and visualization
DeepLog	Anomaly detection model
Transformer networks	Anomaly detection and sequence modeling
Isolation Forest	Unsupervised anomaly detection



2.6 Project Planning

Milestones:

The project will be executed to this timeline, divided into key milestones to ensure timely delivery and effective progress tracking. Each milestone represents a critical phase in the development and deployment of the AI-driven security solution for the CI/CD pipeline.

Milestone 1: Project Initiation and Core Implementation

- Objective: Define project scope and deploy foundational components for anomaly detection and vulnerability scanning.
- Tasks:
  - Finalize project requirements, objectives, and deliverables.
  - Train and deploy anomaly detection model.
  - Integrate vulnerability scanning and code quality enforcement tools.
  - Set up a centralized log management system for data aggregation and visualization.
- Deliverables:
  - Project charter and scope document.
  - Functional anomaly detection model with high accuracy.
  - Integrated vulnerability scanning and code quality tools.
  - Operational log management system for data analysis.

Milestone 2: Advanced Integration and Initial Testing

- Objective: Enhance the system with advanced AI models and conduct initial testing.
- Tasks:
  - Develop and train models for contextual threat classification.
  - Integrate APIs for automated root-cause analysis and incident management.
  - Implement predictive analytics to forecast vulnerabilities from code changes.
  - Conduct initial performance testing for anomaly detection and vulnerability scanning.
- Deliverables:
  - Functional models for contextual threat classification with a reduction in false positives.
  - Operational APIs for incident resolution with low latency.
  - Predictive analytics module for vulnerability forecasting.
  - Initial performance test reports.

Milestone 3: Testing, Optimization, and Deployment

- Objective: Validate system performance, optimize for scalability, and deploy the solution.
- Tasks:
  - Optimize models and tools for scalability and efficiency.
  - Perform user acceptance testing (UAT) with developers, security teams, and administrators.
  - Deploy the final system to production environments.
  - Conduct training sessions for users and provide comprehensive documentation.
- Deliverables:
  - Optimized system with improved performance and scalability.
  - UAT feedback and system refinement.
  - Fully operational system integrated into the CI/CD pipeline.
  - Training materials and user documentation.

Post-Deployment: Continuous Improvement

- Objective: Establish feedback mechanisms and ensure long-term system effectiveness.
- Tasks:
  - Implement feedback loops to refine models and security policies.
  - Monitor system performance and address emerging threats.
  - Apply regular updates and patches to maintain compliance with security standards.
- Deliverables:
  - Feedback mechanisms for continuous improvement.
  - Regular system updates and maintenance reports.

2.7 Financial Projections

Core Tools

Tool/Software	Purpose	Cost (Annual)
Jenkins	CI/CD automation with plugins for AI-powered anomaly detection.	$0 (Open Source)
SonarQube	Static code analysis and code quality enforcement.	$0 (Free)
Kubernetes	Container orchestration	$0 (Open Source)

AI/ML Tools

Tool/Software	Purpose	Cost (Annual)
DeepLog	Deep learning model for anomaly detection in logs.	$0 (Open Source)
Isolation Forest Library	Open-source library for anomaly detection.	$0 (Open Source)
Transformer Networks	Deep Learning architecture for sequential data analysis and anomaly detection.	$0 (Open Source)

Log Management and Monitoring

Tool/Software	Purpose	Cost (Annual)
ELK Stack	Centralized log aggregation, storage, visualization	Free (open-source)

Chapter3: Selection

3.1 Software Technology

1. Data Ingestion and Parsing

Technologies Used: ELK Stack

What it Does:

- Collects logs from operating systems (OS), applications (Apps), and services.

2. Pre-processing & Feature Engineering
   Technologies Used: Regex, Drain3, scikit-learn

   What it Does:

   - Regex: Applies pattern matching to extract and normalize key fields from raw log messages.

   - Drain3: Performs log parsing by clustering similar log messages into templates, converting unstructured logs into structured data.

   - scikit-learn: Extracts and engineers features such as statistical metrics or vector embeddings from structured data to prepare input for AI models.

3. Anomaly Detection Models
   - Technologies Used: PyTorch, Hugging Face, Isolation Forest, LSTM Autoencoders, Transformers
   - What it Does:
     - Runs multiple AI models (e.g., Isolation Forest, LSTM, and Transformer- based models) to detect unusual patterns in the data.

These models work in parallel to improve detection accuracy.

4. Ensemble & Correlation Layer
   - What it Does:
     - Combines results from multiple anomaly detection models to generate a final score.
     - Correlates anomalies with external metrics (e.g., system performance data) to reduce false positives.

5. Feedback Loop & Continuous Learning
   - What it Does:
     - Incorporates human feedback to improve model accuracy over time.
     - Monitors for changes in data patterns (data drift) and automatically retrains models when needed.

6. Monitoring and Visualization
   - Technologies Used: Prometheus, Grafana
   - What it Does:
     - Prometheus collects performance metrics (e.g., CPU usage, memory usage) from systems and applications.
     - Grafana creates dashboards to visualize system health, performance trends, and detected anomalies.

7. Secure CI/CD Pipeline Integration
   - Technologies Used: OWASP, SAST, SonarQube, Trivy, GitOps (ArgoCD), Kubernetes
   - What it Does:
     - Ensures security is integrated into every stage of the software development lifecycle (SDLC). Includes:
       1. Build & Code Coverage: Verifies code quality and test coverage.
       2. SCA/SAST: Scans for vulnerabilities in third-party libraries and code.
       3. Trivy: Scans container images for security issues.
       4. GitOps with ArgoCD: Automates secure deployments on Kubernetes clusters

8. Security Policies
   - Technologies Used: OWASP, Kubernetes Security Standards
   - What it Does:
     - Enforces secure configurations and access controls at every stage of the pipeline.
     - Ensures compliance with OWASP standards and Kubernetes security policies (e.g., Pod Security Standards).

Summary of Key Technologies

Technology	Purpose
Logstash	Parses and standardizes logs for analysis.
Prometheus	Monitors system performance and generates alerts.
Grafana	Visualizes system metrics and anomalies on dashboards.
Regex/Drain3	Converts unstructured logs into structured data.
PyTorch/Hugging Face	Builds and trains AI models for anomaly detection.
Trivy	Scans container images for vulnerabilities.
ArgoCD	Automates secure application deployments using GitOps principles.

3.2 CODE

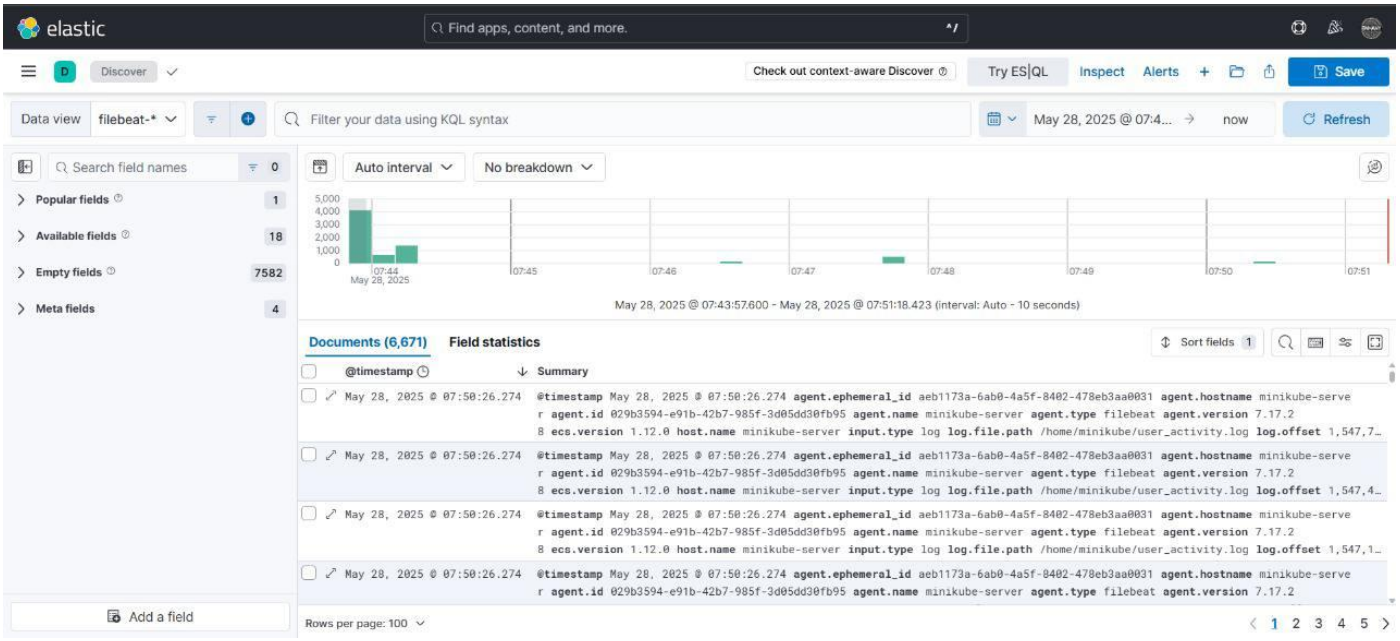


Figure 3. 3 Output of the mode

Chapter 4: Methodology

4.1 System Block Diagram

The system architecture consists of four main components working in an integrated pipeline:

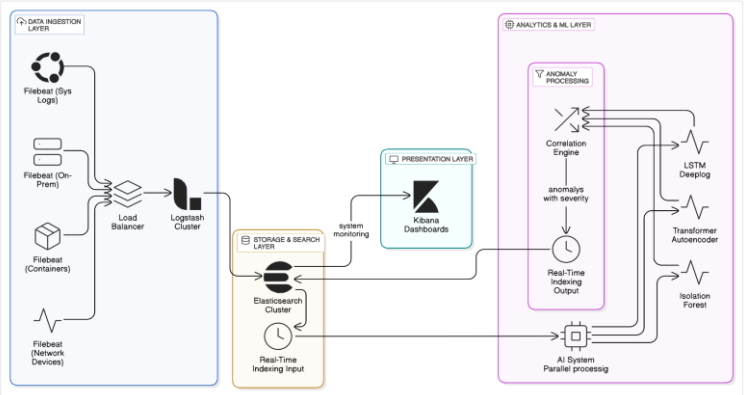


Figure 4.1 System Block Diagram

4.2 Flow Chart of System

The system follows a comprehensive data flow process:

1. Log Collection Phase:
    - Filebeat agents collect logs from various sources
    - Logs are forwarded to Logstash for processing
    - Processed logs are indexed in Elasticsearch
2. Visualization Phase:
    - Kibana provides real-time dashboards and visualizations
    - Users can create custom queries and analytics
3. Anomaly Detection Phase:
    - CSV data export triggers ML model execution
    - Three models process data in parallel using multiprocessing
    - Individual model outputs are generated with confidence scores
4. Correlation Phase:
    - Correlation engine aggregates results from all models
    - Risk assessment algorithm categorizes anomalies
    - Final output provides actionable insights

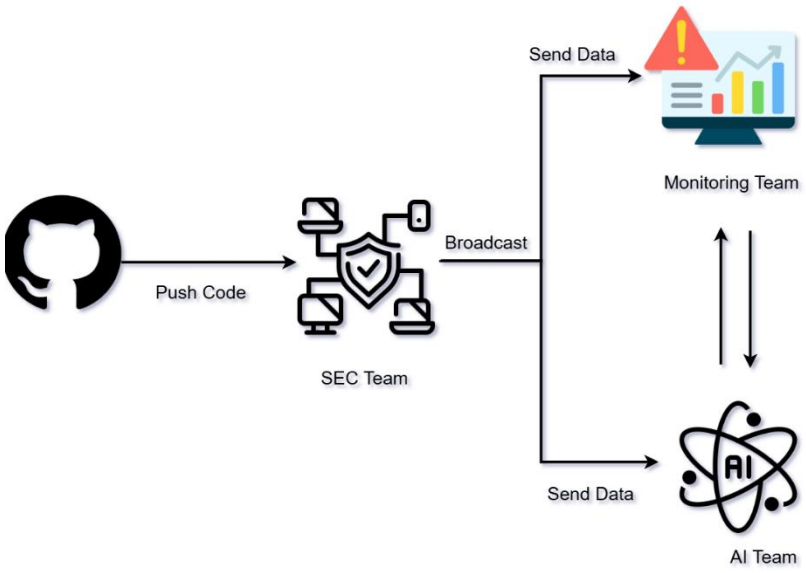


Figure 4.2 Workflow of Project

4.3Architecture Diagram of System

The system architecture emphasizes modularity, scalability, and reliability:

Data Ingestion Layer:

- Multiple Filebeat instances on source systems
- Centralized Logstash cluster for processing
- Load balancing and failover capabilities

Storage and Search Layer:

- Elasticsearch cluster with multiple nodes
- Index lifecycle management for data retention
- Backup and recovery mechanisms

Analytics and ML Layer:

- Parallel model execution framework
- GPU acceleration for neural network models
- Configurable thresholds and parameters

Presentation Layer:

- Kibana dashboards for real-time monitoring
- API endpoints for programmatic access
- Alert mechanisms for critical anomalies

Chapter5: System Implementation

5.1 ELK Stack Implementation

5.1.1 Elasticsearch Configuration

The Elasticsearch installation process involved several critical configuration steps:

Installation Process:

- Add Elastic repository
- Install Elasticsearch

Critical Configuration Parameters:

- Memory Settings: Configured JVM heap size to 50% of available RAM
- Network Configuration: Set network.host and http.port for cluster communication
- Security Settings: Enabled X-Pack security with TLS encryption
- Performance Tuning: Adjusted vm.max_map_count and file descriptor limits

5.1.2 Logstash Pipeline Configuration

Logstash pipeline configuration focused on efficient log parsing and transformation:

- Input Configuration: Configured multiple input sources including Filebeat, TCP, and file inputs
- Filter Configuration: Implemented grok patterns for log parsing, date filtering for timestamp standardization, and mutate filters for field manipulation
- Output Configuration: Configured Elasticsearch output with index templates and error handling

5.1.3 Kibana Dashboard Implementation

Kibana implementation provided comprehensive visualization capabilities:

- Index Pattern Creation: Configured index patterns for log data access
- Dashboard Development: Created custom dashboards for security monitoring
- Visualization Components: Implemented charts, graphs, and tables for data representation
- Alert Configuration: Set up Watcher alerts for critical security events

5.1.4 Filebeat Deployment

Filebeat deployment ensured reliable log collection:

- Module Configuration: Enabled relevant modules for different log types
- Output Configuration: Configured output to Logstash with load balancing
- Security Configuration: Implement

Chapter 6: Conclusion

The documentation presents the design and implementation of a modular, scalable, and intelligent log analysis system. The system architecture integrates four core components log collection, visualization, anomaly detection, and correlation working in a cohesive pipeline. By utilizing the ELK Stack (Elasticsearch, Logstash, Kibana) and Filebeat for log ingestion and visualization, the system ensures efficient data handling and real-time monitoring.

The anomaly detection phase leverages parallel execution of machine learning models with multiprocessing, enhancing detection accuracy and performance. The correlation engine further refines these results by aggregating model outputs and performing risk assessments, enabling the generation of actionable insights.

The architecture is designed with reliability, fault tolerance, and extensibility in mind. Key configurations such as secured Elasticsearch clusters, optimized Logstash pipelines, dynamic Kibana dashboards, and secure Filebeat deployment contribute to the robustness of the system. Overall, this implementation provides a comprehensive, automated, and scalable solution for proactive security monitoring and anomaly detection in modern IT environments.