DEPARTMENT OF INFORMATION TECHNOLOGY
AKHUWAT COLLEGE KASUR
AFFILIATED WITH UNIVERSITY OF THE PUNJAB, LAHORE

FINAL YEAR DESIGN PROJECT

AI-Driven Security Monitoring: Anomaly Detection Using Elastic Stack & Wazuh

AKHUWAT

BS (IT) SESSION:
2021-2025

GROUP MEMEBERS

Khalid Hussain	**(058960)**
Muhammad Osama	**(058949)**

SUPERVISED BY:

EBRYX TEAM

&

MR. MUHAMMAD NAEEM AKHTAR

DECLARATION:

We hereby declare that this Project, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this Project and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other, we will stand by the consequences. No portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

#	Name	Roll no#	Signature:
01	Khalid Hussain	058960	_____
02	Muhammad Osama	058949	_____

CERTIFICATE OF APPROVAL:

It is to certify that the final year design project (FYDP) of BSIT "*AI-Driven Security Monitoring: Anomaly Detection using Elastic Stack & Wazuh*" was developed by Khalid Hussain *(058960)*, Muhammad Osama *(058949)* under the supervision of "Mr. Muhammad Naeem Akhtar", in my opinion it is fully adequate, in scope and quality for the degree of Bachelors of Science in Information Technology.

Name of Supervisor: Mr. Muhammad Naeem Akhtar Signature: _____

Table of Contents

Chapter 1: Proposal

1.1 Project Title

“AI-Driven Security Monitoring: Anomaly Detection using Elastic Stack & Wazuh”

1.2 Project Overview Statement

This project is about creating a system that uses Artificial Intelligence (AI) to detect unusual activities in the Wazuh and Elastic stack Security Suite. These platforms are widely used for managing logs, monitoring file integrity, detecting intrusions, and handling Security Information and Event Management (SIEM).

By adding AI and Machine Learning (ML), the system will improve real-time detection of suspicious behaviors and threats. The system will process logs, show data visually, and use AI to spot anything out of the ordinary. When something unusual is found, it will send alerts with details about the threat level.

Kibana will be the tool for visualizing and analyzing the data in an easy-to-understand way.

1.3 Project Overview Statement Template

Project Title: AI-Driven Security Monitoring: Anomaly Detection using Elastic Stack & Wazuh.

Group Leader: Khalid Hussain

Project Members:

Name	Roll#	Email Address	Signature
Muhammad Osama	058949	osamaacuk.97@gmail.com	
Khalid Hussain	058960	khalidhussain.bsit.auk@gmail.com	

Project Goal: To develop an AI-driven system integrated with Wazuh and elastic stack for real-time anomaly detection, threat alerts, and data visualization.

Objectives:

Sr.#	
1	Integrate AI algorithms with Wazuh and Elastic Stack for anomaly detection.
2	Train AI models to detect odd file creations and too-many logins.
3	Create Kibana dashboards to visualize and explore data easily.
4	Provide real-time alerts with threat levels.
5	Improve security by detecting threats early.

Project Success criteria:

The project will be successful if it can:

- Detect anomalies like anomalous file creation and suspicious login volume with high accuracy.
- Send real-time alerts when unusual activities are found.
- Show clear security insights using Kibana dashboards for easy analysis.
- Work smoothly with Wazuh and the Elastic stack, ensuring reliable performance.
- Improve threat detection by reducing false alarms and identifying real risks.
- Help security teams respond quickly to threats.

Assumptions, Risks, and Obstacles:

- **Assumptions:** We assume the Wazuh and Elastic Stack systems will work well together, there will be enough log data to analyze, AI and machine learning.

• **Risks:** We might face delays due to complex integration, the system's performance could slow down with added processing, there should not be enough logs to train ML model. • **Obstacles:** Learning AI and machine learning might be challenging, poor data quality could affect model accuracy, and we might have limited resources for training and deploying models
Organization Address: This idea is proposed and supervision by **Ebryx.**
Type of project: ❑Research ❑Development
Target End users: • Financial Institutions • Healthcare Organizations • E-commerce Businesses • Government Agencies • Technology Companies • Educational Institutions • Manufacturing & Industrial Companies
Development Technology: ❑Object Oriented ❑Structured
Platform: ❑Web based ❑Distributed ❑Desktop based ❑Setup Configurations ❑Other_____
Project Supervisor: Mr. Muhammad Naeem Akhtar
Approved By:
Date:

1.4 Project Goals & Objectives

Goals

- Develop an AI-powered anomaly detection system integrated with Wazuh and Elastic Stack.
- Enhance real-time threat detection and response for cybersecurity teams.
- Provide actionable insights using Kibana analytics and visualization.

Objectives

- Train and deploy ML models to detect anomalies in log data
- Implement real-time alerts for Anomalous File Creation and Suspicious Logins
- Integrate Kibana dashboards for visualizing common anomalies
- Ensure the system processes logs efficiently and scales with data volumes

1.5 High-level system components

- **Collect & Store Logs:** Using Elastic Stack to gather and save logs.
- **ML Model:** Custom machine learning system to detect anomalies.
- **Alerting System:** Real-time alerts via email or integrations (e.g., Slack)
- **Visualization:** Kibana dashboard.
- **User Interface:** Web-based Kibana interface for analysts

1.6 List of optional functional units

- **Integration with External Security Tools** – Allow compatibility with other security platforms (e.g., Splunk, SIEM).
- **Customizable Alert Rules** – Enable users to modify AI detection thresholds based on their security needs.

1.7 Exclusions

- The system will detect anomalies but will not automatically fix security threats.
- It will not support mobile applications, only a web-based Kibana dashboard.

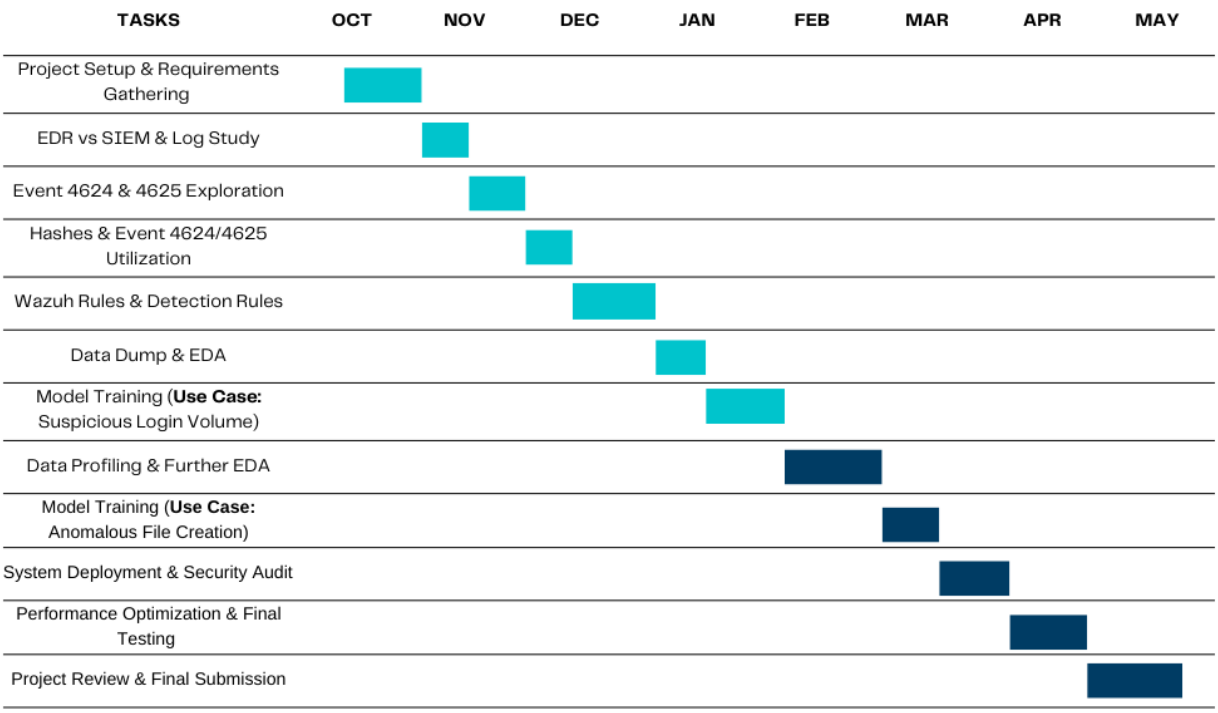
1.8 Application Architecture

The system follows a three-tier architecture:

- **Frontend:** Kibana for interactive data visualization and user dashboards.
- **Backend:** Python-based AI/ML module for log processing, anomaly detection, and alert generation.
- **Database:** Elasticsearch for log storage and fast searching, integrated with Wazuh for security monitoring.

1.9 Gantt Chart

Task	Start Date	End Date	Duration (Days)
Project Setup & Requirements Gathering	17 Oct 2024	31 Oct 2024	15
EDR vs SIEM & Log Study	1 Nov 2024	15 Nov 2024	15
Event 4624 & 4625 Exploration	16 Nov 2024	30 Nov 2024	15
Hashes & Event 4624/4625 Utilization	1 Dec 2024	15 Dec 2024	15
Wazuh Rules & Detection Rules	16 Dec 2024	4 Jan 2025	20
Data Dump & EDA	5 Jan 2025	24 Jan 2025	20
Model Training (Suspicious Login Volume)	25 Jan 2025	15 Feb 2025	22
Data Profiling & Further EDA	16 Feb 2025	5 Mar 2025	18
Model Training (Anomalous File Creation)	6 Mar 2025	26 Mar 2025	21
System Deployment & Security Audit	27 Mar 2025	15 Apr 2025	20
Performance Optimization & Final Testing	16 Apr 2025	5 May 2025	20
Project Review & Final Submission	6 May 2025	31 May 2025	26



1.10 Hardware and Software Specification

Hardware:

- Processor 8GB RAM
- 100GB Storage

Software:

- ELK Stack
- Wazuh
- Python 3.9+ with TensorFlow, scikit-learn
- Kibana
- Filebeat

1.11 Tools and technologies used with reasoning

1. **Elastic Stack & Wazuh** For log management and security monitoring

   - **ELK (Elasticsearch, Logstash, Kibana)** helps collect, store, and analyze logs.
   - **Wazuh** adds security monitoring, helping detect suspicious activities.
   - These tools make it easy to process large amounts of log data and find anomalies.

2. **TensorFlow** For building and training the AI model

   - A powerful open-source machine learning framework.
   - Helps create and train an AI model to detect unusual patterns in data.
   - Supports deep learning techniques, making detection more accurate.

3. **Kibana** For visual analytics and anomaly exploration

   - A dashboard tool that works with Elasticsearch.
   - Helps display logs, trends, and AI-detected anomalies using graphs and charts.
   - Makes it easier to understand and investigate suspicious patterns.

Project Advisor: Mr. Muhammad Naeem Akhtar

Faculty: Department of Information Technology, Akhuwat College Kasur

Chapter 2: Software Requirement Specification

This document outlines the requirements for an AI-powered anomaly detection system built using Wazuh and the Elastic Stack (Elasticsearch, Logstash, Kibana). The system monitors security logs to identify unusual activities, such as:

- **Anomalous File Creation**: Detecting unauthorized or suspicious file creation in restricted directories.
- **Suspicious Login Attempts**: Identifying too many failed login attempts that may indicate brute-force attacks.

The system will use Artificial Intelligence (AI)/Machine Learning (ML) to automatically detect anomalies, provides real-time alerts, and visualizes data on user-friendly Kibana dashboards for security analysts to investigate and respond effectively.

2. Requirements Analysis

This section details the system's requirements, including user needs, constraints, and key features, based on thorough analysis.

2.1 User Classes and Characteristics

User Class	Characteristics
Security Analysts	Monitor logs, review alerts, investigate incidents, and take corrective actions.
System Administrators	Configure and maintain Wazuh and Elastic Stack components, manage system uptime.
AI/ML Engineers	Develop, train, and optimize AI/ML models for anomaly detection.

2.2 Requirement Identification Techniques

Requirements were gathered using the following methods:

- **Use Case Analysis**: Developed detailed use cases and diagrams to map system interactions and features, ensuring alignment with user needs.
- **Stakeholder Interviews**: Consulted security teams and IT staff to understand practical security challenges and system expectations.
- **Data-Driven Analysis**: Examined real-world security logs to define performance for AI/ML anomaly detection.
- **Literature Review**: Studied research papers on AI-based security systems to adopt best practices and address potential challenges.

3. Functional Requirements

3.1 System Features

The system provides the following core functionalities:

ID	Feature	Description
FR-1	Log Collection	Collects security logs from devices and applications in real-time.
FR-2	AI-Based Anomaly Detection	Uses AI/ML to identify unusual patterns in logs (e.g., anomalies).
FR-3	Alert System	Sends real-time alerts when threats or anomalies are detected.
FR-4	Dashboard Visualization	Displays security data on Kibana dashboards with graphs and filters.
FR-5	Anomalous File Creation Detection	Detects and alerts on unauthorized or suspicious file creation events.
FR-6	Suspicious Login Attempt Detection	Monitors and alerts on too many failed login attempts within a short period.
FR-7	Alert Customization	Allows users to set custom alert thresholds and response actions.
FR-8	Incident Reporting	Enables logging and documentation of incidents for analysis and compliance.

3.2 Detailed Functional Requirements

Identifier	**FR-1**
Title	Log Collection
Requirement	The system shall collect security logs from various devices and applications.
Source	Need for security monitoring from endpoint.
Rationale	Helps detect security threats by analyzing logs in one place.
Business Rule	Alerts should be generated if a file is created in a restricted directory or by an unauthorized user.
Dependencies	None
Priority	High

Identifier	**FR-2**
Title	AI-Based Detection
Requirement	The system shall use AI to detect unusual activities in collected logs.
Source	Requirement for advanced threat detection.
Rationale	Identifies hidden threats not caught by standard rules.
Business Rule	The AI model should analyze logs in real-time and flag unusual patterns.
Dependencies	FR-1 *(Log Collection)*
Priority	High

Identifier	**FR-3**
Title	Alert System
Requirement	The system shall send alerts when a potential security threat is detected.
Source	Requirement for quick threat notification.
Rationale	Enables fast response to potential security issues.
Business Rule	Alerts should be sent via email and displayed on the dashboard.
Dependencies	FR-2 *(AI-Based Detection)*
Priority	High

Identifier	**FR-4**
Title	Dashboard
Requirement	The system shall present security data in a clear and easy-to-understand format.
Source	Need for accessible security monitoring.
Rationale	Allows security teams to monitor the system effectively.
Business Rule	The dashboard should display alerts, logs, and system status.
Dependencies	FR-1 *(Log Collection)*, FR-3 *(Alert System)*
Priority	High

Identifier	**FR-5**
Title	Anomalous File Creation Detection
Requirement	The system shall detect and alert users about unusual file creation activities based on AI analysis.
Source	Security monitoring needs from Wazuh logs.
Rationale	Helps security analysts detect potential security threats in real time.
Business Rule	Alerts should be generated if a file is created in a restricted directory or by an unauthorized user.
Dependencies	FR-1 *(Log Collection)*, FR-2 *(AI-Based Detection)*
Priority	High

Identifier	**FR-6**
Title	Suspicious Login Attempt Detection
Requirement	The system shall monitor and identify **too many failed** login attempts from a single user or IP address within a short time frame.
Source	Security policy for monitoring unauthorized access attempts.
Rationale	Prevents brute-force attacks and helps in identifying compromised accounts.
Business Rule	The system should generate alerts if failed login attempts 5 within a minute.
Dependencies	FR-1 *(Log Collection)*, FR-3 *(Alert System)*
Priority	High

Identifier	**FR-8**
Title	Incident Reporting
Requirement	The system shall enable security analysts to log incidents and document investigation details.
Source	Need for proper incident management and documentation
Rationale	Supports compliance and helps improve future security practices.
Business Rule	Incident reports should include the incident's date, time, actions taken, and resolution.
Dependencies	FR-4 (Dashboard)
Priority	High

4. Non-Functional Requirements

4.1 Reliability

- The system shall achieve 99.9% uptime, excluding planned maintenance.
- It shall handle up to 500 MB of daily logs without performance degradation.
- Automated backups shall prevent data loss in case of failure.

4.2 Speed and Performance

- Security logs shall be processed and indexed within 5 seconds.
- Log search queries shall return results within 2 seconds.
- AI-based alerts shall be generated within 1 second of anomaly detection.

4.3 Compatibility

- The system shall support Windows, Linux, and macOS log sources.
- It shall integrate with third-party tools (e.g., Splunk) via APIs.
- Log formats (e.g., Syslog, JSON) shall be supported without manual configuration.

4.4 Scalability

- The system shall process logs from 10–50 endpoints, with the ability to scale to 100+.
- It shall maintain performance with up to 15 GB of monthly logs.
- Alerts shall remain reliable with up to 100 concurrent notifications.

4.5 Maintainability

- Updates shall be applied without downtime using rolling updates.
- Bugs shall be fixed within 24 hours of detection during development.
- Comprehensive documentation shall guide troubleshooting and maintenance.

5. External Interfaces

5.1 User Interface

- Kibana dashboards shall include:
  - Graphs (e.g., login attempt trends, file creation events).
  - Filters (e.g., by time, IP, user).
  - Tables summarizing alerts and incidents.
- Alerts shall be sent via email with clickable links to dashboard details or shown in dashboard.

5.2 Software Interfaces

- **Wazuh**: Collects and processes logs from endpoints.
- **Elastic Stack**: Stores (Elasticsearch), processes (Filebeat), and visualizes (Kibana) logs.
- **AI/ML Models**: Python-based models (e.g., scikit-learn) analyze logs for anomalies.

5.3 Hardware Requirements

- The system needs at least 8GB RAM and 100GB storage.

5.4 Communication

- The system shall use secure HTTPS/TLS for web-based access to Kibana.
- Log data shall be transmitted via encrypted channels (e.g., Wazuh agent protocols).

6. Use Case Analyses

6.1 Use Case 1: Anomalous File Creation Detection

Field	Details
UC Identifier	UC-1
Requirements Traceability	FR-5 (Anomalous File Creation Detection)
Purpose	Detect and alert on unauthorized file creation in restricted directories.
Priority	High
Preconditions	Wazuh agents are monitoring file system logs.
Postconditions	Alerts are generated and logged for analyst review.
Actors	Security Analysts, System Administrators
Extends	None
Main Success Scenario	**1.** System monitors file creation logs. **2.** AI detects a file created in a restricted path (e.g., /etc). **3.** Alert is sent via email and dashboard. **4.** Analyst investigates and deletes the file if malicious.
Alternate Flows	Analyst marks the file as safe if it's legitimate.
Exceptions	If AI fails, Wazuh's default rules trigger alerts.
Includes	FR-1 (Log Collection), FR-2 (AI-Based Detection), FR-3 (Alert System)

6.2 Use Case 2: Suspicious Login Attempt Detection

Field	Details
UC Identifier	UC-2
Requirements Traceability	FR-6 (Suspicious Login Attempt Detection)
Purpose	Detect and alert on too many failed login attempts.
Priority	High
Preconditions	Authentication logs are being collected.
Postconditions	Alerts are triggered if 5+ failed logins occur within 1 minute.
Actors	Security Analysts, System Administrators
Extends	None
Main Success Scenario	1. System tracks login attempts.2. AI detects 5+ failed logins from an IP.3. Alert is sent via email and dashboard.4. Analyst blocks the IP if suspicious.
Alternate Flows	Analyst dismisses the alert if logins are valid.
Exceptions	Network issues log failures, and alerts are queued.
Includes	FR-1 (Log Collection), FR-2 (AI-Based Detection), FR-3 (Alert System)

7. Use Case Diagram

The use case diagram ***(Figure 01)*** illustrates the interactions between actors (Security Analysts, System Administrators) and use cases (UC-1, UC-2). It shows how analysts monitor alerts and administrators configure the system.

- **Actors**: Security Analyst, System Administrator.
- **Use Cases**: Anomalous File Creation Detection, Suspicious Login Attempt Detection.
- **Relationships**: Analyst interacts with both use cases **,** Administrator configures system settings**.**

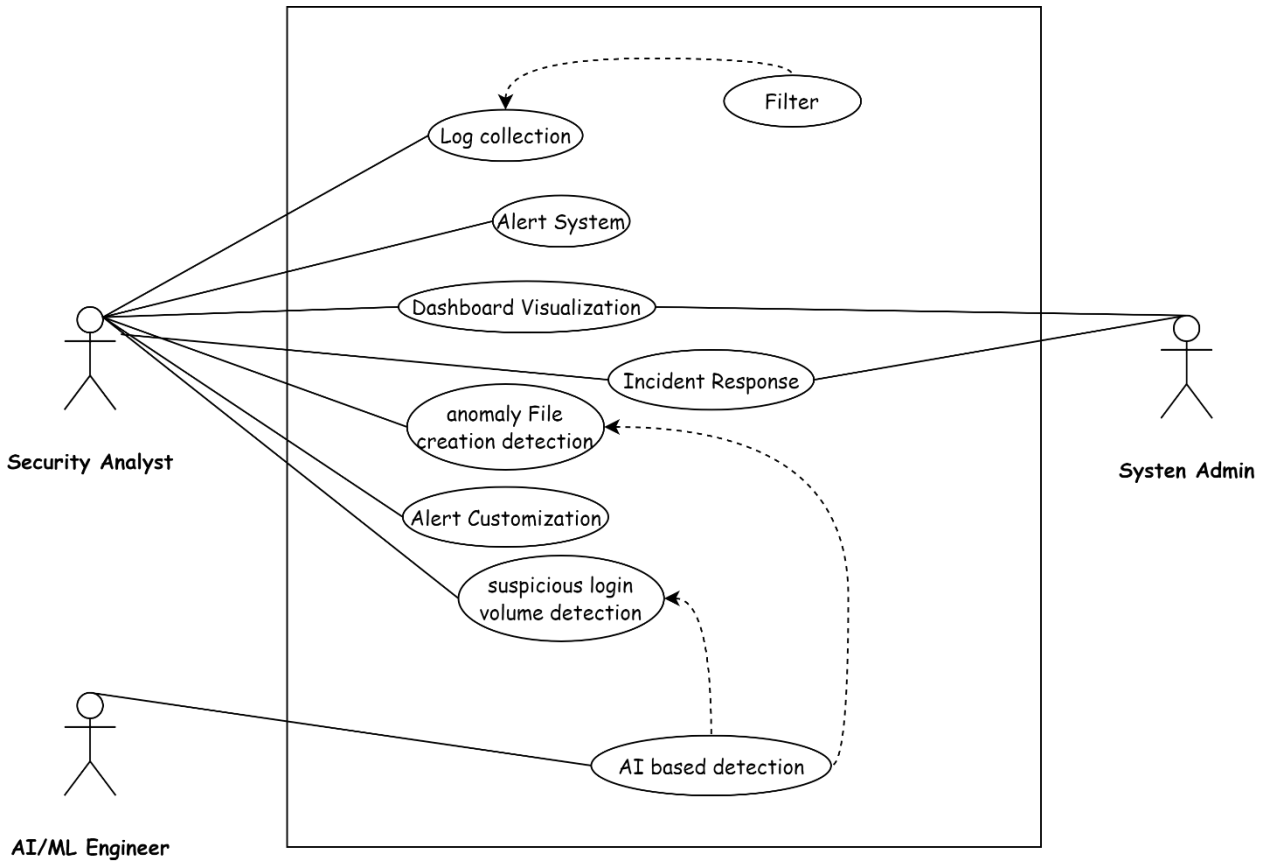


Figure: 01

9. Summary

This document describes an AI-powered anomaly detection system using Wazuh and the Elastic Stack to identify unusual activities, such as *anomalous file creation* and *suspicious login attempts*. The system uses Artificial intelligence for automatic detection, real-time monitoring, and instant alerts, with a clear visualization of data on Kibana dashboards.

Designed for security analysts, system administrators, and AI engineers, it offers features like log collection, customizable alerts, and incident reporting. It supports multiple platforms, scales with growing data, and integrates with other security tools. The system provides clear dashboards and email notifications to keep users informed.

Two key use cases detecting *unusual file creation* and *suspicious logins* demonstrate how it helps security teams respond quickly and effectively

Chapter 3: Software Design Specification

1. Product Perspective

The system is a security monitoring tool that integrates Wazuh (a Security Information and Event Management platform) with the Elastic Stack (Elasticsearch, Logstash, Kibana) to collect, analyze, and visualize security logs. It uses AI-driven anomaly detection to identify threats like unauthorized file creation and suspicious login attempts, providing real-time alerts and dashboards for security teams.

2. Design Considerations

- **Assumptions**:
    - Wazuh, Filebeat, and Elastic Stack are correctly installed and configured.
    - Sufficient log data is available for AI/ML analysis.
    - Users have basic familiarity with Kibana dashboards.
- **Dependencies**:
    - Wazuh for log collection and initial processing.
    - Elastic Stack for storage, processing, and visualization.
    - Python libraries (e.g., scikit-learn) for AI/ML models.
- **Limitations**:
    - Alerts are sent but no automatic threat mitigation (e.g., auto-blocking IPs).
    - Web-based Kibana dashboards only, no mobile app support.
- **Risks and Mitigation**:
    - **Integration Challenges**: Issues connecting to external log sources.
        - **Mitigation**: Test integrations with common log formats (e.g., Syslog, JSON) during development and provide setup guides.
    - **Performance Issues**: Slowdowns with large log volumes (>15 GB monthly).
        - **Mitigation**: Optimize Elasticsearch indexing and scale with additional nodes for larger deployments.
    - **Data Quality**: Poor log data reducing AI accuracy.
        - **Mitigation**: Preprocess logs with filebeat to ensure consistency and validate AI models with diverse datasets.

3. Requirements Traceability Matrix

The matrix links requirements from the SRS to design components and test cases, ensuring all functional and non-functional requirements are addressed.

Requirement ID	Description	Design Component	Test Case	Implementation Status
FR-1	Log Collection	Wazuh Agents, Filebeat	Verify logs collected from 10 endpoints	Completed
FR-2	AI-Based Anomaly Detection	AI/ML Model (Python)	Test anomaly detection accuracy (>90%)	In Progress
FR-3	Alert System	Wazuh Manager, Kibana	Confirm alerts sent within 1 second	Completed
FR-4	Dashboard Visualization	Kibana Dashboards	Validate dashboard updates in real-time	Completed
FR-5	Anomalous File Creation Detection	Wazuh FIM, AI Model	Test alerts for unauthorized file creation	In Progress
FR-6	Suspicious Login Attempt Detection	Wazuh, AI Model	Test alerts for too many failed logins in a minute	In Progress
FR-7	Alert Customization	Kibana Alert Rules	Verify custom threshold settings	Not Started
FR-8	Incident Reporting	Kibana Reporting	Test report generation and export	Completed
NFR-1	Reliability (99.9% uptime)	Elasticsearch Clustering	Monitor uptime over 30 days	Completed

Requirement ID	Description	Design Component	Test Case	Implementation Status
NFR-2	Ease of Use (In-built UI)	Kibana Interface	Usability test with 5 analysts	Not Started
NFR-3	Speed (5-second log processing)	Elasticsearch	Measure log processing time	In Progress

4. Design Models

4.1 Architectural Design

The system uses a **Multi-Tier Architecture** with the following layers:

- **Data Collection Layer**:
    - **Tool**: Wazuh Agents
    - **Purpose**: Collects logs from endpoints (e.g., servers, workstations).
    - **Function**: Monitors file changes, login attempts, and system events.
- **Processing Layer**:
    - **Tool**: Wazuh Manager
    - **Purpose**: Filters and analyzes raw logs.
    - **Function**: Applies security rules and prepares logs for forwarding.
- **Log Forwarding Layer**:
    - **Tool**: Filebeat
    - **Purpose**: Transfers logs to Elasticsearch.
    - **Function**: Ensures efficient, reliable log delivery.
- **Storage Layer**:
    - **Tool**: Elasticsearch
    - **Purpose**: Stores and indexes logs.
    - **Function**: Enables fast search and retrieval for analysis.
- **Analysis Layer**:
    - **Tool**: AI/ML Model (Python-based, e.g., isolation forests)
    - **Purpose**: Detects anomalies like unauthorized file creation or too many login attempts.
    - **Function**: Analyzes logs for unusual patterns with high accuracy.
- **Visualization Layer**:
    - **Tool**: Kibana
    - **Purpose**: Displays logs, alerts, and reports.
    - **Function**: Provides interactive dashboards for security analysts.

4.2 Data Design

Elasticsearch manages all log data, optimized for fast search and analysis. Key data structures include:

- **Log Data**:
  - **Purpose**: Tracks system activities (e.g., file creation, logins).
  - **Example**:
```
{  "timestamp": "2025-04-24T10:00:00",
   "event": "file_created",
   "path": "/etc/config",
   "user": "guest" }
```
- **Alert Data**:
  - **Purpose**: Stores details of detected threats.
  - **Example**:
```
{ "alert_id": "A001",
  "type": "suspicious_login",
  "source_ip":"192.168.1.10",
  "attempts": 6,
  "time": "2025-04-24T10:01:00" }
```
- **User Data**:
  - **Purpose**: Manages access roles.
  - **Example**:
```
{  "user_id": "U001",
   "role": "analyst",
   "permissions":
  ["view_alerts", "generate_reports"]}
```

Data Dictionary

Term	Description	Constraints
Log Event	Records system activities (e.g., file changes, logins).	Must include timestamp, event type, source.
Anomaly Alert	Details detected threats (e.g., alert type, source).	Must include alert ID, timestamp.
User Role	Defines user access levels (e.g., analyst, admin).	Must specify role and permissions.

4.3 User Interface Design

The Kibana dashboard provides an intuitive interface for security analysts:

- **Alert Panel**:
  - Displays alerts with threat levels (e.g., low, high), timestamps, and sources.
  - Allows marking alerts as “safe” or “threat” with one click.
- **Log View**:
  - Shows filterable log entries in a table (columns: time, event, source, details).
  - Includes search bar for quick queries (e.g., “failed login”).

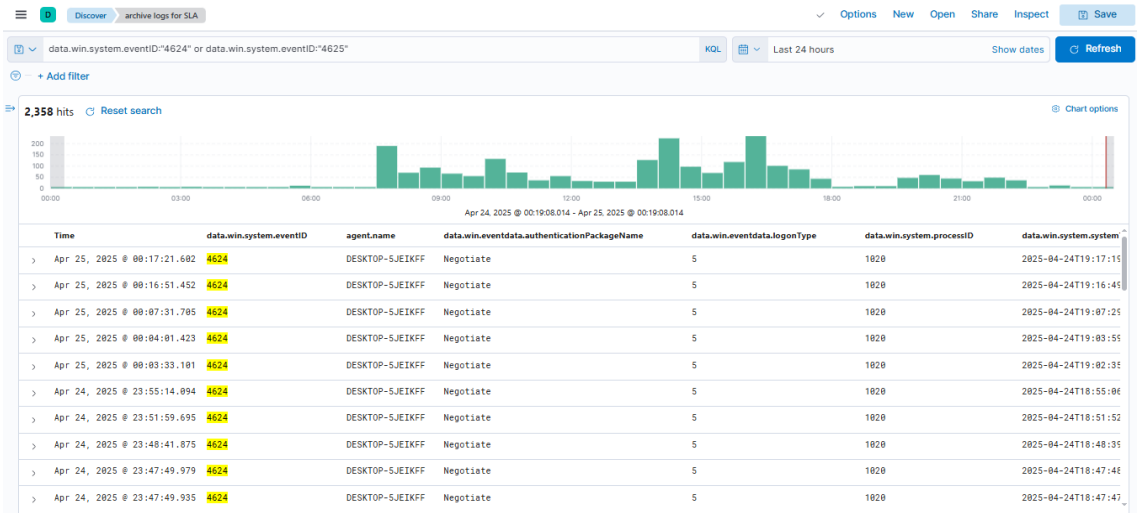


Figure 01

- **Graphs**:
  - Visualizes trends (e.g., login attempts by IP, file creation by directory).
  - Supports time-range filters (e.g., last 24 hours or more).

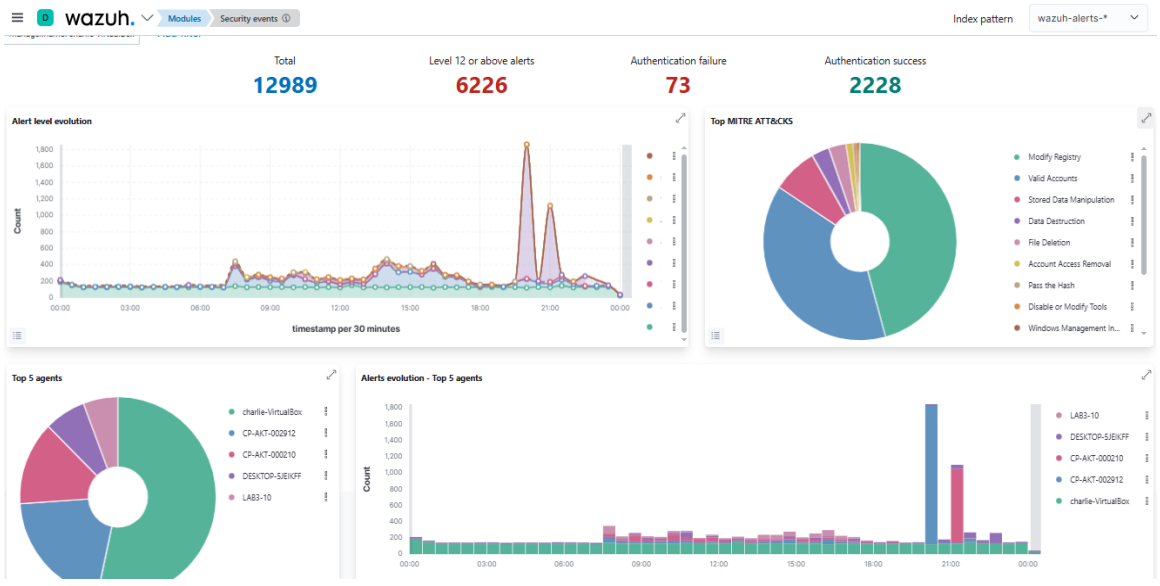


Figure 02

4.4 Behavioral Model

The system operates in a sequential workflow:

1. **Log Collection**: Wazuh agents gather logs from endpoints.
2. **Data Processing**: Wazuh Manager filters logs; Filebeat sends them to Elasticsearch.
3. **Anomaly Detection**: AI/ML model analyzes logs for anomalies
4. **Alert Generation**: Alerts are created and sent via email or displayed on Kibana.
5. **User Action**: Analysts review alerts, investigate, and log incidents.

Interaction Diagrams

- **Sequence Diagram** (*Figure 03*): Illustrates log flow:
    - **Actors**: Endpoint, Wazuh Agent, Wazuh Manager, Filebeat, Elasticsearch, AI Model, Kibana, Analyst.
    - **Flow**:
        1. Endpoint sends log to Wazuh Agent.
        2. Agent forwards to Wazuh Manager.
        3. Manager processes and sends to Filebeat.
        4. Filebeat stores in Elasticsearch.
        5. AI Model analyzes and detects anomaly.
        6. Alert is sent to Kibana and emailed.
        7. Analyst views alert and responds.

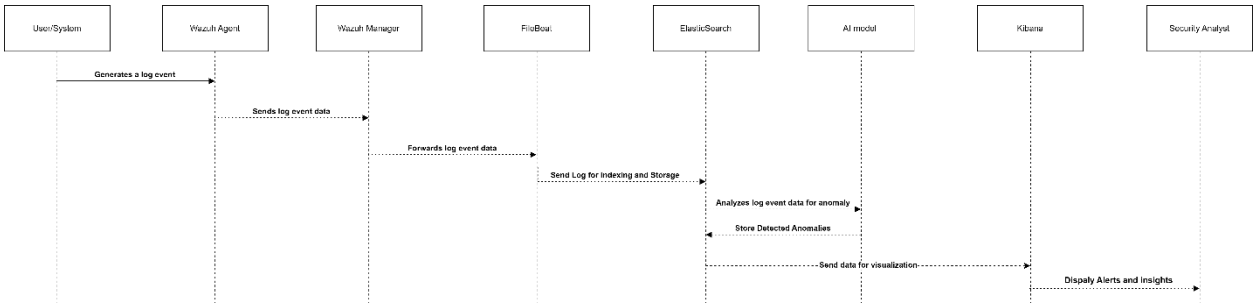


Figure 03

5. Design Decisions

- **AI Model**:
    - **Choice**: Unsupervised ML (e.g., isolation forests).
    - **Reason**: Detects anomalies without labeled data, ideal for diverse log patterns.
- **Data Storage**:
    - **Choice**: Elasticsearch.
    - **Reason**: Fast indexing and search for large log volumes.
- **Frontend**:
    - **Choice**: Kibana.
    - **Reason**: Provides user-friendly dashboards with real-time visualization.

6. Summary

This system combines Wazuh, Elastic Stack, and AI to monitor security logs and detect threats like unauthorized file creation and suspicious logins. Its multi-tier architecture ensures efficient log collection, processing, and visualization. Kibana dashboards provide clear insights, and AI reduces false positives. Designed for 10–50 endpoints, it scales with additional nodes for larger networks, keeping systems secure with fast, reliable alerts.

7. References

- *Wazuh* All-in-one deployment with Elastic Stack. *https://documentation.wazuh.com/4.5/deployment-options/elastic-stack/all-in-onedeployment/index.html*
- Machine learning & security protecting systems with data and algorithms by *clarencechio & david freeman https://a.co/d/cFCmldy*

- ***Security Monitoring with Wazuh:*** A hands-on guide to effective enterprise security using real-life use cases in Wazuh by *Rajneesh Gupta https://a.co/d/540PA1L*

Chapter 4: Implementation

This chapter explains how we built our project, "AI-Driven Security Monitoring: Anomaly Detection with ELK Stack and Wazuh." It covers the main parts of the system, including the machine learning algorithm used to detect threats, the tools and technologies we integrated, and how we organized and managed the project code. Each section gives a step-by-step explanation of how different components were set up and made to work together.

4.1 Algorithm

At the core of our system is the **Isolation Forest** machine learning algorithm, which we use to detect unusual or suspicious activities in log data. These anomalies can include things like multiple failed login attempts or files being created in folders where they shouldn't be. One big advantage of Isolation Forest is that it doesn't need labeled data to work. Instead of being told what is "normal" or "suspicious," it learns patterns from the data on its own and identifies anything that stands out.

How It Works

The algorithm works by trying to separate each log entry from the rest of the data. If a log looks very different like a login at a strange time or a file being created in a restricted folder it becomes easier to spot. If the algorithm can isolate it quickly, it marks it as an anomaly, meaning it might be suspicious or unusual activity.

Integration with Elasticsearch

To make this work in real time, we've connected our machine learning model with Elasticsearch. Here's the simplified workflow:

1. **Log Data Collection**: We start by gathering log data, which could include login attempts, file creations, or other system activities.

2. **Data Preparation**: The collected data is then cleaned and formatted to ensure it's suitable for analysis. This step involves handling missing values and ensuring proper data formatting.

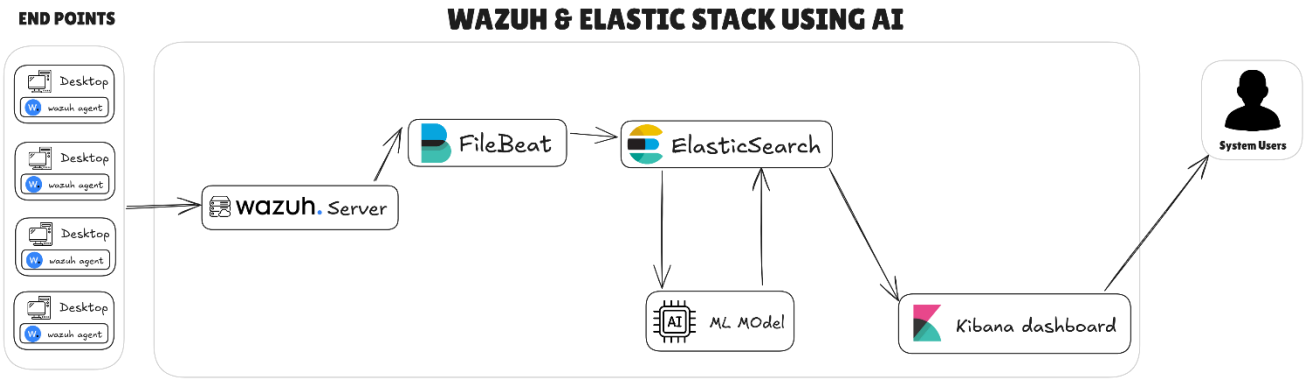
3. **Model Training**: Using historical log data, we train the Isolation Forest model to recognize normal patterns and identify deviations.

4. **Real-Time Analysis**: For each new log entry:

- The model analyzes the log.

- If it detects something suspicious, it triggers an alert.
- This alert is then sent back to Elasticsearch.

5. **Dashboard Visualization**: The alerts are displayed in real time on Kibana dashboards, providing security analysts with immediate insights and the ability to take swift action.



(This figure Shows how system works)

4.2 External APIs and Libraries

To build our system, we used several tools and libraries, each with a special role:

- **Scikit-learn**: This library helped us use the Isolation Forest algorithm. It gave us the tools to train the machine learning model and make predictions.
- **Wazuh API**: We used this to collect logs and get information about connected agents. It made it easy for our Python scripts to work with Wazuh data.
- **Elasticsearch API**: This allowed us to search through log data and save alerts created by the AI model. It also helped connect our model to the Kibana dashboard for visual display.
- **Matplotlib and Seaborn**: These were used to create graphs and charts during training and testing. They helped us understand how well the model was working and make improvements.

Chapter 5: Testing and Evaluation

Making sure our project works correctly and reliably was very important. In this chapter, we explain the different types of testing we did to check our system. This includes unit testing, functional testing, integration testing, and performance testing. The main goal was to make sure every part of the system works properly and that everything runs smoothly when used in real situations.

5.1 Unit Testing (UT)

Unit testing involved examining individual components of our project in isolation. Here are some of the test cases we conducted:

Testcase ID	**UT-01**
Requirement ID	FR-1
Title	Log Collection
Description	Test whether Wazuh collects logs from endpoints
Objective	Ensure log collection module is working
precondition	Wazuh agent installed
Test steps	**1.** Start agent **2.** Trigger activity **3.** Check Elasticsearch
Input	login attempt
Expected Result	Logs appear in index
Actual Result	Success
Remarks	Pass

Testcase ID	**UT-02**
Requirement ID	FR-2
Title	AI Model Test
Description	Check ML model scores input log correctly
Objective	Ensure model detects anomaly
precondition	Model trained
Test steps	**1.** Feed abnormal log **2.** Run model **3.** Check result
Input	Anomalous log entry
Expected Result	Low score (anomaly)
Actual Result	Detected
Remarks	Pass

Testcase ID	**UT-03**
Requirement ID	FR-3
Title	Alert Creation Logic
Description	Test internal logic that generates alerts
Objective	Ensure alert logic functions
precondition	ML score available
Test steps	**1**. Trigger low-score event **2.** Alert function runs
Input	Alert threshold met
Expected Result	Alert created
Actual Result	Success
Remarks	Pass

Testcase ID	**UT-04**
Requirement ID	FR-7
Title	Custom Alert Settings
Description	Test if alerts respect new threshold
Objective	Confirm user-defined settings apply
precondition	Access to rule config
Test steps	**1.** Lower threshold **2.** Trigger event
Input	3 failed logins
Expected Result	Alert triggered early
Actual Result	Success
Remarks	Pass

5.2 Functional Testing (FT)

Functional testing focused on validating whether the system's features met the specified requirements. We checked if the system could accurately detect anomalies like suspicious file writes and brute-force login attempts. For instance:

Testcase ID	**FT-01**
Requirement ID	FR-3
Title	Alert Delivery
Description	Test if alerts appear in Kibana and email
Objective	Ensure alerts reach the analyst
precondition	ML alert created
Test steps	**1.** Trigger anomaly **2.** View alert in dashboard
Input	Unusual login
Expected Result	Alert visible
Actual Result	Visible
Remarks	Pass

Testcase ID	**FT-02**
Requirement ID	FR-4
Title	Dashboard Graphs
Description	Check if Kibana shows alerts and logs correctly
Objective	Ensure data is easy to view
precondition	Kibana configured
Test steps	**1.** Open dashboard **2.** Apply filters **3.** Review charts
Input	Sample queries
Expected Result	Visual data shown

Actual Result	Working as expected
Remarks	Pass

Testcase ID	**FT-03**
Requirement ID	FR-5
Title	File Creation Alert
Description	Alert on new file in restricted path
Objective	Detect unauthorized file creation
precondition	FIM enabled
Test steps	**1.** Create file in /etc **2.** Watch for alert
Input	touch /etc/test.txt
Expected Result	Alert triggered
Actual Result	Success
Remarks	Pass

Testcase ID	**FT-04**
Requirement ID	FR-6
Title	Brute Force Login
Description	Detect too many failed logins
Objective	Alert if login failure exceeds limit
precondition	Authentication logs monitored
Test steps	**1.** Try 6 bad logins **2.** Watch Kibana
Input	Suspicious attempts
Expected Result	Alert visible
Actual Result	Yes
Remarks	Pass

Testcase ID	**FT-05**
Requirement ID	FR-7
Title	Custom Rules
Description	Verify custom thresholds
Objective	Detect early when threshold lowered
precondition	Config updated
Test steps	**1.** Lower rule **2.** Trigger alert
Input	3 failed logins
Expected Result	Alert early
Actual Result	Works
Remarks	Pass

Testcase ID	**FT-06**
Requirement ID	FR-8
Title	Incident Notes
Description	Save response taken by analyst
Objective	Record for future reference
precondition	Alert exists
Test steps	**1.** Open alert **2.** Add note
Input	Blocked IP
Expected Result	Note saved
Actual Result	Confirmed
Remarks	Pass

5.3 Integration Testing (IT)

Testcase ID	**IT-01**
Requirement ID	FR-2, FR-3, FR-4
Title	AI to Dashboard Flow
Description	Check ML alert reaches Elasticsearch and Kibana
Objective	End-to-end anomaly detection to alert visualization
precondition	All services connected
Test steps	**1.** Trigger anomaly **2.** Send alert **3.** View Kibana
Input	Suspicious event
Expected Result	Alert shown
Actual Result	Success
Remarks	Pass

5.4 Performance Testing (PT)

Performance testing allowed us to assess the system's speed and stability, especially when handling large volumes of data. Our tests included:

Testcase ID	**PT-01**
Requirement ID	FR-1
Title	Index Speed
Description	Check time to index logs
Objective	Confirm logs added in < 5s
precondition	Log file ready
Test steps	**1.** Upload file **2.** Measure
Input	500MB JSON logs
Expected Result	≤ 5s
Actual Result	4.2s
Remarks	Pass

Testcase ID	**PT-02**
Requirement ID	FR-2
Title	Model Speed
Description	Measure model delay
Objective	Confirm detection under 1s
precondition	Model loaded
Test steps	**1.** Send log **2.** Measure
Input	Log with anomaly
Expected Result	< 1s
Actual Result	0.6s
Remarks	Pass

Testcase ID	**PT-03**
Requirement ID	FR-3
Title	Alert Delay
Description	Measure time from anomaly to alert view
Objective	Under 2 seconds total
precondition	ML alert triggered
Test steps	**1.** Trigger alert **2.** View in Kibana
Input	Detected log
Expected Result	Alert in < 2s
Actual Result	1.1s
Remarks	Pass

5.5 Summary

Throughout this chapter, we've demonstrated how we tested our system at various levels:

- **Unit Testing**: We verified the functionality of individual components like log collection and machine learning detection.
- **Functional Testing**: We ensured that the system could detect anomalies such as suspicious logins and file changes as required.
- **Integration Testing**: We confirmed that there was smooth communication and data flow between Wazuh, Elasticsearch, our machine learning model, and Kibana.
- **Performance Testing**: We validated that the system could process logs quickly and maintain stability even when handling large datasets.

Chapter 6: System Conversion

Transitioning from the old way of doing things to our new software system required careful planning and execution. This chapter details the system conversion process, including the method we chose, the deployment steps, data conversion, training, post-deployment testing, and the challenges we faced.

6.1 Conversion Method

We chose the **Phased Conversion** method, which means we added the new system step by step instead of all at once. This made it easier to find and fix any problems early, without affecting the whole system. Here’s how we did it, one part at a time.Here's the sequence we followed:

1. **Log Collection Module**: We started by deploying the log collection module, ensuring that logs could be gathered from various endpoints.
2. **Anomaly Detection**: After setting up log collection, we added the part that finds unusual activities using our machine learning model.
3. **Alerting and Dashboards**: Finally, we enabled the alerting system and set up the Kibana dashboards for visualization and real-time monitoring.

6.2 Deployment

The deployment process involved several critical steps to ensure the system was set up correctly:

1. **Setting Up the Server**: We began by installing Ubuntu 25.04 LTS on the target server, providing a stable foundation for our system.
2. **Installing and Configuring Tools**: We then installed and configured essential tools such as Wazuh, Filebeat, Elasticsearch, and Kibana. Each of these tools plays a specific role in log collection, processing, and visualization.
3. **Deploying AI Models**: Using Python and required libraries like scikit-learn, we deployed our machine learning models. This step was crucial for enabling the anomaly detection capabilities of our system.
4. **Connecting Services**: We connected all the services and tested their communication to ensure data could flow seamlessly through the system.

5. **Starting Services and Verifying Functionality**: With everything connected, we started the services and verified that real-time data collection and alerting were working as expected.

6. **Accessing Kibana Dashboard**: Finally, we accessed the Kibana dashboard through a browser to begin monitoring the data and ensuring everything was functioning correctly.

6.2.1 Data Conversion

Although our system wasn't replacing an existing one, we still needed to prepare log data for use:

- **Extracting Logs**: We extracted logs from test systems, including server login activities and file operations.
- **Cleaning and Formatting**: The logs were cleaned and formatted into JSON or Syslog format to ensure consistency and compatibility.
- **Loading into Elasticsearch**: Using Filebeat and Wazuh Manager, we loaded the logs into Elasticsearch.

  Testing Data Integrity: We compared sample input logs with the output alerts to verify that the data had been correctly processed and that the alerts were accurate.

6.2.2 Training

To ensure that security analysts and administrators could effectively use the system, we created a basic user manual. This manual included training on specific use cases:

- **File Creation Alert**: Users were trained to open the Kibana dashboard, check the alert panel for new file creation alerts, view details such as file path, user, and time, and take appropriate action like marking the alert as safe or escalating it.
- **Suspicious Login Attempt**: We taught users to navigate to the "Login Attempts" dashboard, use filters to identify failed logins by IP or user, recognize unusual activity, and take action such as blocking the source IP if necessary and logging the incident.

6.3 Post Deployment Testing

After the system was deployed, we carried out a series of tests to ensure everything was working correctly:

- **AI Detection Testing**: We simulated attacks to see if the AI could detect them and generate appropriate alerts.
- **Alert Generation Time**: We checked that alerts were generated within one second of detecting an anomaly, ensuring real-time responsiveness.
- **Dashboard Updates**: We reviewed the dashboards to confirm they were updating in real time with the latest information.
- **Email Alert Verification**: We confirmed that alerts were being sent via email as expected, providing an additional notification channel.
- **System Health Checks**: We ran comprehensive health checks to confirm the system's uptime and resource usage, ensuring it was operating efficiently and reliably.

6.4 Challenges

During the deployment process, we encountered several challenges that required our attention:

- **Integration Issues**: We faced some problems with the integration between Wazuh and Elasticsearch. By carefully reviewing the configuration files, we were able to resolve these issues.
- **Slow Response Time**: When dealing with large logs, we noticed a slower response time. Optimizing the Elasticsearch index settings helped improve performance.
- **AI Model Tuning**: Inconsistent data made it difficult to tune the AI model. We addressed this by adding data preprocessing steps to ensure the data fed into the model was consistent and of high quality.

Chapter 7: Conclusion

7.1 Evaluation

We set out with several objectives in Chapter 1, and we're pleased to report that we've successfully completed them all. Here's a recap:

- **Log Collection**: We've implemented log collection from multiple endpoints using Wazuh and Filebeat.
- **Anomaly Detection**: Our AI/ML models are effectively detecting anomalies in log data.
- **Real-Time Alerts**: The system is now capable of showing real-time alerts for suspicious activities like unusual file creation and login attempts.
- **Dashboard Visualization**: Security logs are being visualized on interactive Kibana dashboards, providing users with an intuitive way to monitor their systems.
- **Alert Customization and Incident Reporting**: We've enabled features for customizing alerts and generating incident reports, giving users more control over how they're notified and how they document security events.
- **Performance**: The system meets our performance goals, with log processing occurring in under five seconds.

7.2 Traceability Matrix

To ensure comprehensive coverage, we mapped each requirement from the Software Requirements Specification (SRS) to its corresponding design component, module, and test ID. This traceability matrix served as a valuable tool for tracking how each requirement was implemented and verified.

Requirement ID	Description	Design Component	Code/Module	Test ID
FR-1	Log Collection	Wazuh Agents, Filebeat	log_collector.py	UT1
FR-2	AI-Based Anomaly Detection	AI Model (Python)	anomaly_detector.py	UT2
FR-3	Alert System	Kibana, Wazuh Manager	alert_module.py	FT1
FR-4	Dashboard Visualization	Kibana Dashboards	dashboard_config.json	FT2
FR-5	File Creation Detection	AI Model, FIM Module	file_watch.py	FT3
FR-6	Login Attempt Detection	AI Model, Auth Logs	login_watch.py	FT4
FR-7	Alert Customization	Kibana Rules	developed	FT6
FR-8	Incident Reporting	Kibana Reports	report_module.py	FT5

7.4 Future Work

While we're satisfied with the system's current capabilities, there are several areas where we can continue to improve and expand its functionality:

- **AI Accuracy Improvement**: By training our models on larger and more diverse datasets, we can further enhance the accuracy of our AI-driven threat detection.
- **Automatic Threat Mitigation**: We're exploring the addition of automatic threat mitigation features, such as blocking IPs after repeated failed login attempts, to make the system more proactive in responding to threats.
- **Mobile-Friendly Dashboard**: Developing a mobile-friendly version of the dashboard will allow security personnel to monitor their systems on the go, providing greater flexibility and responsiveness.
- **Advanced Customization Options**: We plan to add more customizable reports and alert filters to cater to the needs of advanced analysts who require more detailed and specific information.
- **Third-Party Integrations**: Integrating with third-party tools like Splunk or other SIEM solutions via API will expand the system's capabilities and allow it to fit more seamlessly into existing security infrastructures.

By pursuing these future enhancements, we aim to continue evolving our AI-driven security monitoring system, making it even more powerful and adaptable to the ever-changing landscape of cybersecurity threats.