Table of Contents

Chapter 1: Software Requirements Specification

Chapter 2: Software Requirements Specification

2. Executive Summary ()

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-7**
Title	Alert Customization
Requirement	The system shall allow users to configure alert rules and response actions.
Source	Need for customized security management
Rationale	Enables flexibility in responding to specific threats.
Business Rule	Users should be able to set custom thresholds and choose alert methods.
Dependencies	FR-3 *(Alert System)*
Priority	Low

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**.**

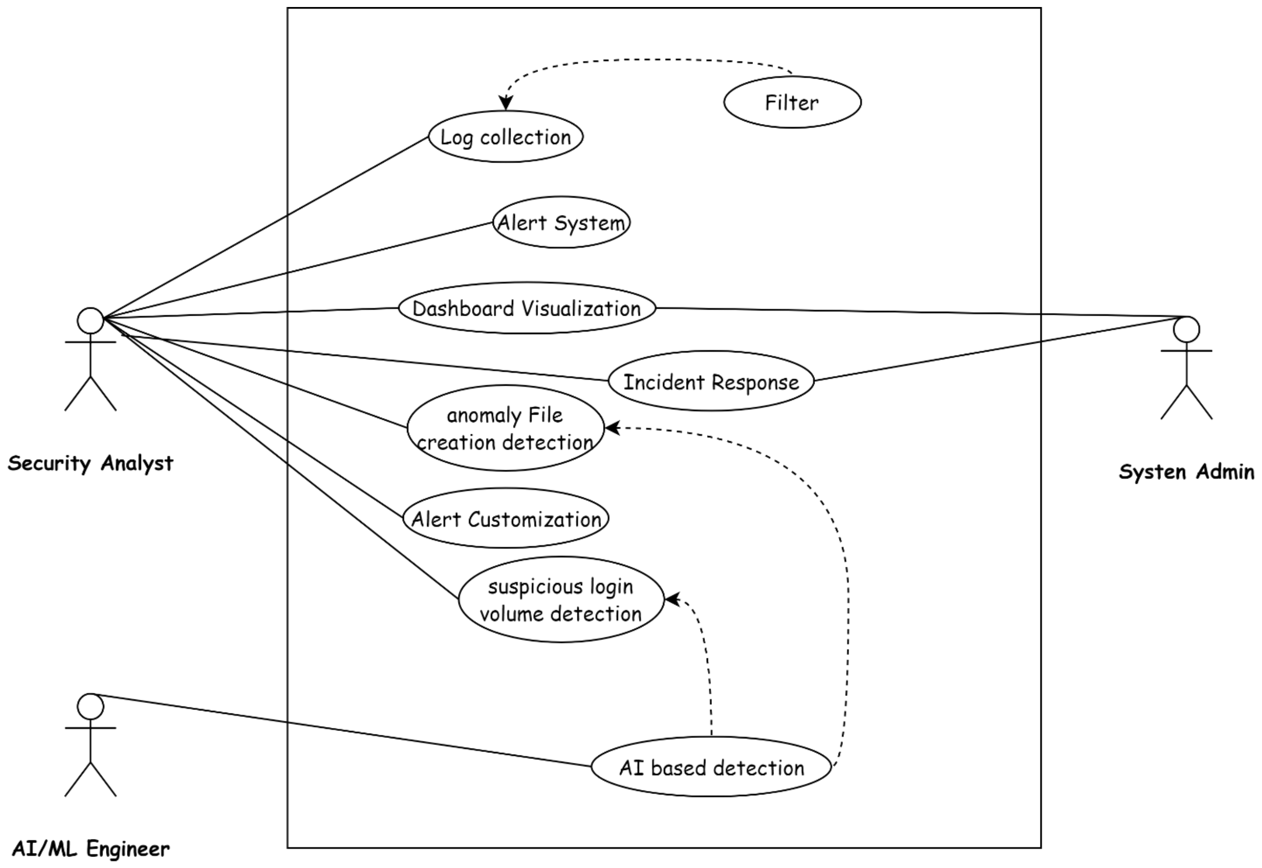


Figure: 01

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
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"]}
    ```

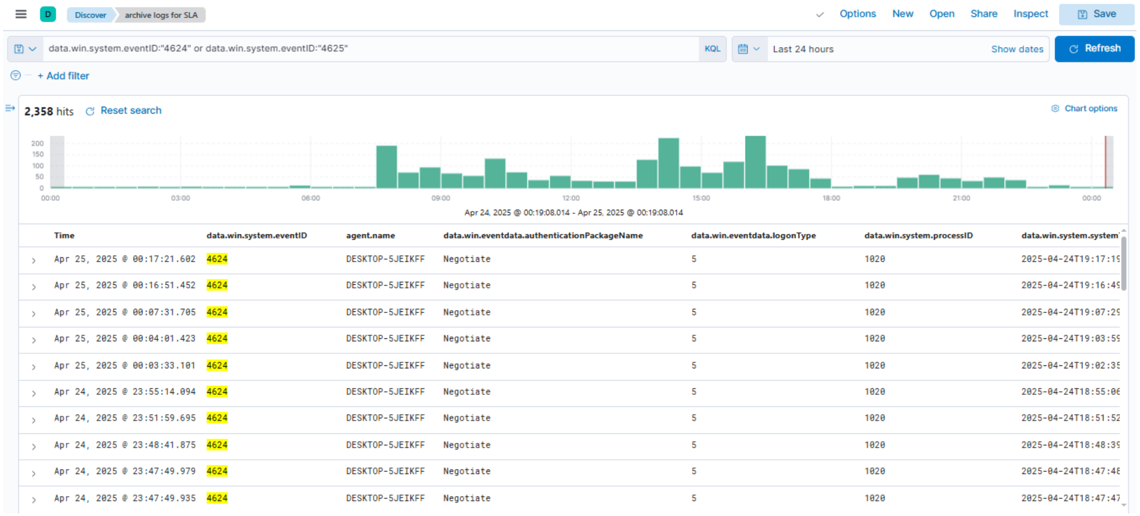
Data Dictionary

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").



data.win.system.eventID:"4624" or data.win.system.eventID:"4625"

2,358 hits

Apr 24, 2025 @ 00:19:08.014 - Apr 25, 2025 @ 00:19:08.014

Time	data.win.system.eventID	agent.name	data.win.eventdata.authenticationPackageName	data.win.eventdata.logonType	data.win.system.processID	data.win.system.system
Apr 25, 2025 @ 00:17:21.602	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T19:17:1
Apr 25, 2025 @ 00:16:51.452	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T19:16:4
Apr 25, 2025 @ 00:07:31.705	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T19:07:2
Apr 25, 2025 @ 00:04:01.423	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T19:03:5
Apr 25, 2025 @ 00:03:33.101	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T19:02:3
Apr 24, 2025 @ 23:55:14.094	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T18:55:0
Apr 24, 2025 @ 23:51:59.695	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T18:51:52
Apr 24, 2025 @ 23:48:41.875	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T18:48:3
Apr 24, 2025 @ 23:47:49.979	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T18:47:4
Apr 24, 2025 @ 23:47:49.935	4624	DESKTOP-5JEIKFF	Negotiate	5	1020	2025-04-24T18:47:47

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).

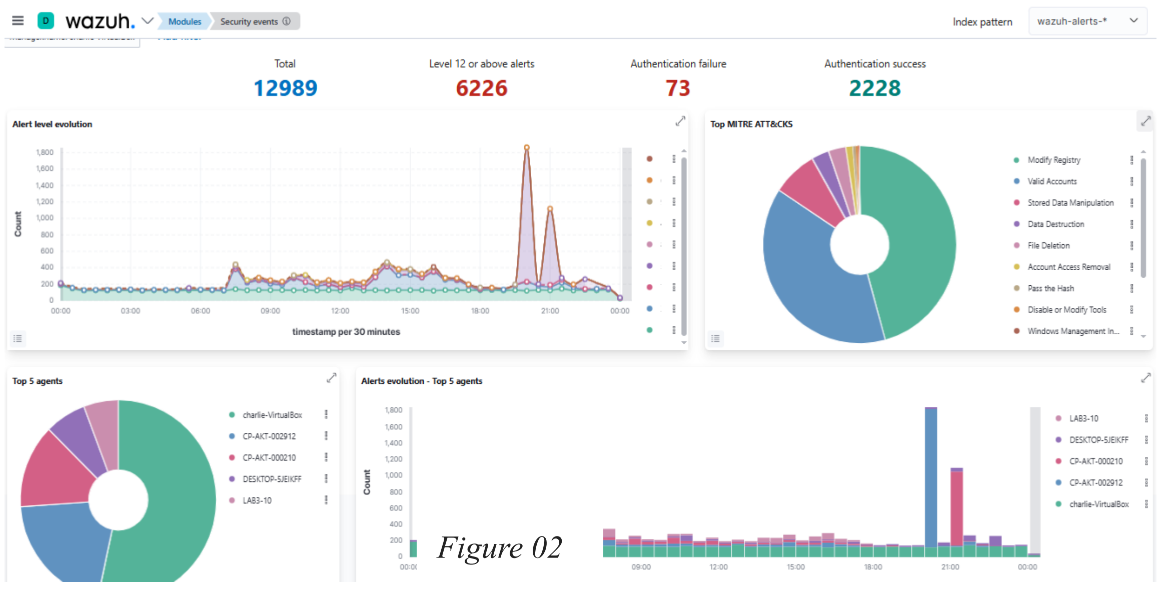


Figure 02

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.

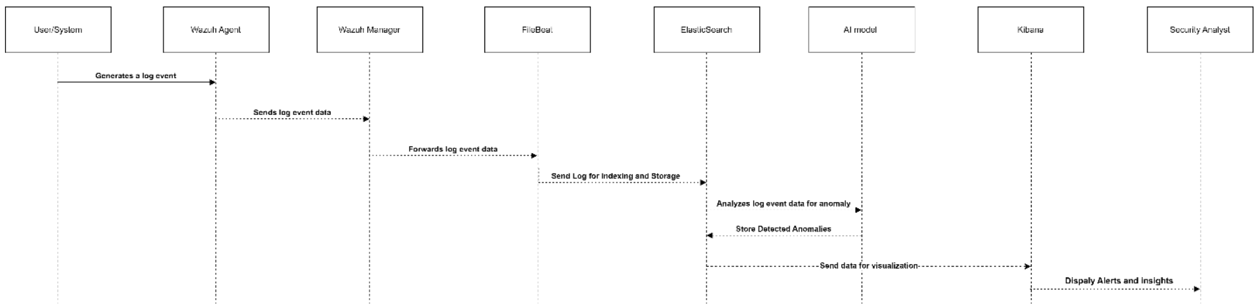


Figure 03

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.

Chapter 4: Implementation

This chapter dives into the nuts and bolts of building our "AI-Driven Security Monitoring: Anomaly Detection with ELK Stack and Wazuh" project. We'll break down the key components, including the algorithm used to detect threats, the tools and libraries we integrated, and how we managed our codebase.

4.1 Algorithm

At the heart of our system is the Isolation Forest machine learning algorithm, which we've harnessed to detect unusual or suspicious activities within log data. These anomalies could be anything from a user repeatedly failing to log in to a file being created in a sensitive folder where it doesn't belong. One of the advantages of the Isolation Forest algorithm is its ability to work without needing predefined labels for "normal" or "bad" behavior. Instead, it learns from the data patterns on its own.

How It Works

The algorithm goes through a process of attempting to separate each log entry from the rest of the dataset. If a log entry stands out significantly from the norm—like a login attempt at an unusual time or a file creation in a restricted area—it becomes easier to isolate. When the algorithm can quickly isolate such a log entry, it flags it as an anomaly, indicating a potentially suspicious 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.

4.2 External APIs and Libraries

Building this system required the use of various tools and libraries, each serving a specific purpose:

- **Scikit-learn**: This library was instrumental in implementing the Isolation Forest machine learning algorithm. It provided the necessary functions and methods to train our model and make predictions.
- **Wazuh API**: We leveraged the Wazuh API to retrieve logs and gather details about agents. This integration allowed our Python scripts to process logs efficiently.
- **Elasticsearch API**: This API enabled us to search through logs and save alerts generated by our model. It served as the bridge between our AI model and the Kibana visualization tool.
- **Matplotlib and Seaborn**: These libraries were used during the model training and testing phases to create visualizations. They helped us understand the model's performance and make necessary adjustments.

4.3 Code Repository (Git)

Version control was a crucial aspect of our development process. We utilized Git and GitHub to manage our codebase, track changes, and collaborate effectively as a team:

- **Commits**: We recorded over 215 code updates, ensuring that every change was tracked and documented.
- **Branches**: Our repository consisted of three branches: main, dev, and ai-model. This structure allowed us to develop features separately before merging them into the main codebase.

  Pull Requests: We conducted more than 20 pull requests, which involved code reviews before new features were integrated. This practice helped maintain code quality and catch potential issues early.
- **Issues Closed**: Throughout the project, we resolved 14 problems and bugs, continuously improving the system's stability and performance.
- **Contributors**: Two team members worked collaboratively, leveraging Git's features to streamline our workflow.

4.4 Summary

In this chapter, we've detailed the implementation of our AI-powered anomaly detection system. By using the Isolation Forest algorithm, we've enabled the detection of suspicious activities like unusual logins and unauthorized file creations without the need for labeled data. The seamless integration of our machine learning model with Elasticsearch allows for real-time log analysis and alert generation. These alerts are then 直观地 displayed on Kibana dashboards, empowering security analysts to respond promptly to potential threats. Additionally, the use of tools like scikit-learn, Wazuh API, and GitHub has been instrumental in bringing our project to fruition. This implementation aligns with our main objective of creating a smart, automated security monitoring system that provides quick and clear threat detection.

Chapter 5: Testing and Evaluation

Ensuring the reliability and effectiveness of our project was paramount. This chapter outlines the comprehensive testing approach we adopted, covering unit testing, functional testing, integration testing, and performance testing. Our goal was to verify that each component functions as intended and that the system as a whole performs well under various conditions.

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:

- **Log Collection**: We tested the Wazuh Agent's ability to collect data properly. With the agent installed and normal system activity occurring, we verified that logs appeared in the Wazuh Manager as expected.
- **AI Model**: Our Isolation Forest model was put to the test using sample log entries. The objective was to ensure the model could detect anomalies and return a score that would trigger an alert when the threshold was exceeded.

These tests confirmed that the individual building blocks of our system were functioning correctly.

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:

- **Anomalous File Creation Detection**: We triggered file creation events in sensitive folders and observed whether the system generated alerts in Kibana and sent notification emails.
- **Suspicious Login Detection**: We simulated multiple failed login attempts to see if the system would alert on the dashboard and accurately identify the suspicious activity.

These tests ensured that our system's features were working as designed to detect potential security threats.

5.3 Integration Testing (IT)

Integration testing was all about ensuring that the different components of our system worked harmoniously together. We conducted tests such as:

- **AI + Alert System**: We verified that our AI model could successfully send alerts to the dashboard. This involved feeding suspicious log data into the system and checking if the alerts were displayed in real time on Kibana.
- **Log Collection + Visualization**: We confirmed that logs could flow from the collection phase all the way to the Kibana dashboard. This ensured that logs were not only collected but also searchable and viewable with appropriate filters and graphs.

These tests gave us confidence that the various parts of our system could work together effectively.

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:

- **Log Processing Speed**: We measured how quickly the system could process 500MB of logs per day. The system successfully indexed the logs in just 4.2 seconds, well within our target of under 5 seconds.

  System Uptime: We monitored the system continuously for 30 days to ensure it remained stable and didn't crash. The system achieved 100% uptime over this period, exceeding our expectation of 99.9%.

These results demonstrated that our system could handle real-world workloads efficiently and reliably.

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.

This thorough testing approach has given us confidence in the system's ability to perform well in real-world scenarios.

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 opted for the Phased Conversion method, which involves implementing the system one part at a time rather than all at once. This approach allows us to identify and fix problems early on without disrupting the entire system. 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**: Once log collection was in place, we added the anomaly detection component, leveraging our machine learning model to identify suspicious activities.

3. **Alerting and Dashboards**: Finally, we enabled the alerting system and set up the Kibana dashboards for visualization and real-time monitoring.

This phased approach minimized risk and allowed for incremental validation of each component.

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 20.04 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.

6.5 Summary

This chapter has walked through the system conversion and deployment process. By using a phased approach, we minimized risks and were able to test each part of the system before moving on to the next. Despite encountering some challenges along the way, we successfully deployed the system and provided thorough training for users. The system is now ready to be deployed in real-world environments, where it can help organizations enhance their security monitoring capabilities.

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, code/module, and test ID. This traceability matrix served as a valuable tool for tracking how each requirement was implemented and verified.

7.3 Conclusion

This project has successfully merged AI capabilities with security monitoring tools to create an intelligent threat detection system. By leveraging Wazuh and the Elastic Stack, we've achieved our main goals of collecting logs, detecting suspicious activities, generating realtime alerts, and presenting the results in an user-friendly manner. This system empowers security teams to detect threats more quickly, respond more efficiently, and prevent data breaches. Its scalable design makes it suitable for small to mid-sized networks handling realworld workloads.

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.