CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

As cyber threats grow in complexity and frequency, organizations must adopt proactive and automated security strategies to safeguard their digital assets. Traditional Security Operations Centers (SOCs) often rely on manual processes for threat detection and incident response, which can lead to delayed reactions, increased false positives and analyst fatigue. To address these challenges, this project explores the integration of two open-source platforms—Wazuh and Shuffle—to build a real-time, automated threat monitoring and response system.

Complementing Wazuh, **Shuffle** acts as a low-code Security Orchestration, Automation and Response (SOAR) engine. It enables the creation of modular workflows that automate repetitive tasks such as alert enrichment, threat intelligence lookups and remediation actions. By integrating Shuffle with Wazuh's alerting pipeline, the system can respond to threats in real time—executing predefined playbooks that isolate compromised assets, notify stakeholders and initiate recovery procedures.

Wazuh serves as the backbone of the monitoring infrastructure, offering comprehensive capabilities in log aggregation, file integrity monitoring, intrusion detection, vulnerability assessment and compliance reporting. It collects and analyzes data from endpoints, servers, cloud workloads and network devices, generating alerts based on customizable rules and threat signatures.

This project aims to demonstrate how open-source tools can be orchestrated to deliver enterprise-grade security automation with minimal overhead. The architecture emphasizes scalability, modularity and transparency, allowing security teams to adapt workflows to evolving threat landscapes. Key components include RESTful API integrations, real-time dashboards and approval gates to ensure human oversight in critical response actions.

Through simulated attack scenarios and performance benchmarks, the project validates the system's ability to detect threats, reduce response latency and improve operational efficiency. The outcome is a robust framework that enhances an organization's cybersecurity posture while reducing manual workload and accelerating incident resolution.

1.2 OBJECTIVE:

In today's rapidly evolving cyber threat landscape, organizations require robust, scalable and automated solutions to detect, analyze and respond to security incidents in real time. This project aims to design and implement an integrated **Security Information and Event Management (SIEM)** and **Security Orchestration, Automation and Response (SOAR)** framework using **Wazuh** and **Shuffle**, enabling proactive threat monitoring and automated incident response.

The primary objective is to build a cost-effective, open-source-driven security operations solution that empowers security teams to detect anomalies, correlate events across distributed systems and initiate timely responses with minimal human intervention. Wazuh, a powerful SIEM platform, will serve as the core for log collection, threat detection, file integrity monitoring and compliance auditing. It will be configured to ingest logs from multiple endpoints, servers and cloud services, normalizing and analyzing them in real time using rule-based detection mechanisms.

To complement Wazuh's detection capabilities, **Shuffle** will be integrated as the SOAR layer to automate incident response workflows. Shuffle's low-code/no-code interface allows for the creation of dynamic playbooks that can trigger actions such as isolating compromised hosts, sending alerts to security analysts, updating ticketing systems, or even blocking malicious IPs via firewall APIs. This integration ensures that once a threat is detected by Wazuh, Shuffle can immediately act upon it, reducing mean time to respond (MTTR) and minimizing potential damage.

The project will also focus on building a modular and extensible architecture that supports scalability and customization. Key components include:

- **Log Forwarding & Parsing**: Using agents like Wazuh and Filebeat to forward logs from endpoints and cloud services.
- **Threat Intelligence Integration**: Enriching alerts with external threat feeds (e.g., AbuseIPDB, VirusTotal).
- **Custom Detection Rules**: Tailoring Wazuh rules to detect specific attack patterns such as brute-force attempts, privilege escalation, or lateral movement.
- **Automated Playbooks**: Designing Shuffle workflows for common incident types (e.g., malware detection, unauthorized access, suspicious outbound traffic).
- **Dashboarding & Alerting**: Visualizing security events via Kibana and configuring alert thresholds for critical incidents.

By the end of the project, the system will be capable of:

- Real-time detection of security threats across heterogeneous environments.
- Automated triage and response actions based on predefined playbooks.
- Centralized visibility into security posture with actionable insights.
- Reduced analyst fatigue through automation of repetitive tasks.

This solution not only demonstrates the power of open-source tools in building enterprise- grade security infrastructure but also provides a practical, hands-on framework for organizations seeking to enhance their cyber resilience. It serves as a blueprint for integrating detection and response capabilities in a cohesive, automated and intelligent manner.

CHAPTER 2
LITERATURE REVIEW

2.1 Existing Survey:

TITLE: AUTOMATED DEFENSE AGAINST APPLICATION-LAYER ATTACKS ON WINDOWS SYSTEMS USING WAZUH AND SHUFFLE
AUTHORS: AASTHA THAKKER, KAPIL KUMAR, 2025

Application-layer attacks targeting Windows systems remain a significant threat due to their ability to bypass traditional perimeter defenses. These attacks often exploit vulnerabilities listed in the OWASP Top 10 for desktop applications, demanding proactive defense mechanisms. This paper proposes a unified approach that combines SIEM and SOAR capabilities to detect and respond to Windows-based application-layer threats with increased efficiency and automation. The framework integrates the open-source SIEM platform Wazuh with the SOAR engine Shuffle to automate threat detection and incident response.

TITLE: SECURITY AND THREAT DETECTION THROUGH CLOUD-BASED WAZUH DEPLOYMENT
AUTHORS: SYED MOIZ, ABDUL MAJID, 2024

This research emphasizes the critical need to safeguard organizational assets from cyber threats and highlights the pivotal role that firewalls play in fortifying network and cyber systems. The challenge lies in maintaining updated firewall defenses to prevent server downtime and the circumvention of security measures. To address these concerns, this study introduces a novel approach by integrating Cloud Wazuh, a user-friendly solution tailored for non-technical personnel, providing streamlined deployment and customizable security rules. Leveraging Wazuh suite of tools, particularly the Host-Based Intrusion Detection System (HIDS), the research detects and mitigates potential threats, including patterns associated with attacks such as Port Scanning, Denial of Service (DoS) and Metasploit.

TITLE: IMPROVING THREAT DETECTION IN WAZUH USING MACHINE LEARNING TECHNIQUES
AUTHORS: SAMIR ACHRAF CHAMKAR, MOUNIA ZAYDI, 2025

The escalating complexity, frequency and scale of cyber threats present growing challenges to the effectiveness of traditional Security Operations Centers (SOCs). As cyberattacks evolve beyond signature-based exploits to include sophisticated, stealthy and adaptive techniques, SOCs must adopt intelligent, scalable and real-time threat detection mechanisms to maintain organizational resilience and response capability. Wazuh, a widely adopted open-source Security Information and Event Management (SIEM) system, plays a central role in providing security visibility through the collection and correlation of logs, intrusion alerts and system anomalies across distributed infrastructure.

TITLE: **REAL-TIME DEFENSEAGAINSTCYBER THREATS: ANALYZING WAZUH'S EFFECTIVENESS IN SERVER MONITORING**
AUTHORS: ALDE ALANDA, H.A MOODUTO, RONAL HADI, 2023

As cloud computing grows exponentially, organizations face escalating cybersecurity challenges due to increased cyber threats and attacks on cloud-based networks. Monitoring cloud servers is one action that can be taken to improve the security. This can be done with the help of various server monitoring tools, such as Wazuh. The study investigates Wazuh's effectiveness in real-time monitoring of three AWS EC2 instance-based cloud servers. Wazuh's capabilities such as log data collection, malware detection, active response automation and Docker container monitoring, are examined

TITLE: SIEM AND THREAT INTELLIGENCE: PROTECTING APPLICATIONS WITH WAZUH AND THEHIVE
AUTHORS:JUMIATY, BENFANO SOEWITO

This paper explores the use of Wazuh in conjunction with TheHive for application-level threat detection. It emphasizes the importance of real-time alerting and correlation of events across distributed systems. While Shuffle is not directly used, the study lays foundational insights into integrating SIEM with SOAR platforms for automated response

TITLE: SECURITY OPERATIONS CENTER USING OPEN-SOURCE TOOLS
AUTHORS: KUNAL WALAVALKAR, KEDAR WALAVALKAR, YASH J AGDALE, 22024

This research presents a full-stack open-source SOC architecture using Wazuh, Shuffle, TheHive and Cortex. It details the deployment pipeline, alert triaging and response orchestration. The paper highlights Shuffle’s role in automating playbooks triggered by Wazuh alerts, showcasing a practical implementation of real-time threat response.

CHAPTER 3
SYSTEM ANALYSIS

3.1 Existing System:

The existing system relies on manual log analysis, isolated security tools and delayed incident response, making it inefficient against modern cyber threats. Logs are scattered across endpoints with no centralized SIEM and alerts often lack context due to missing threat intelligence. Response actions are manual, increasing mean time to detect and respond. There's limited scalability, no automation and high analyst fatigue due to repetitive tasks and alert overload. Without orchestration, security operations remain reactive and fragmented. This setup fails to meet compliance standards and leaves organizations vulnerable to multi-stage attacks, data breaches and operational disruption. A smarter, automated solution is needed.

3.1.1 Drawbacks:

1. **Manual Response Workflow**

   Security analysts must manually investigate and respond to alerts, leading to delays and inconsistent remediation.

2. **Lack of Centralized Visibility**

   Logs and events are scattered across endpoints, making it difficult to correlate incidents and detect multi-stage attacks.

3. **No Real-Time Detection**

   Traditional systems often process logs in batches, resulting in delayed threat identification and slower response times.

4. **High Alert Fatigue**

   Excessive false positives overwhelm analysts, reducing efficiency and increasing the risk of missing critical threats.

5. **Limited Scalability**

   Adding new devices or services requires manual configuration, making it hard to adapt to growing infrastructure.

6. **No Threat Intelligence Integration**

   Alerts lack enrichment from external sources, reducing context and accuracy in threat classification

3.2 Proposed System:

The proposed system integrates Wazuh (SIEM) and Shuffle (SOAR) to deliver a real-time, automated threat monitoring and response framework. Wazuh collects and analyzes logs from endpoints, servers and cloud services, detecting anomalies using custom rules and threat intelligence feeds. Shuffle complements this by executing automated playbooks that isolate threats, notify analysts and trigger remediation actions. This integration reduces response time, enhances visibility and minimizes manual effort. The system is scalable, open-source and tailored for proactive security operations. It empowers organizations to detect, correlate and respond to threats swiftly while maintaining compliance and operational resilience across diverse environments.

3.2.1 Advantages:

- **Real-Time Threat Detection**

  Wazuh continuously monitors logs and system activity, enabling immediate identification of suspicious behavior and anomalies.

- **Automated Incident Response**

  Shuffle executes predefined playbooks to respond to threats instantly—isolating hosts, sending alerts, or blocking IPs without manual intervention.

- **Centralized Visibility**

  Security events from multiple sources are aggregated and visualized in a unified dashboard, improving situational awareness.

- **Scalability & Flexibility**

  The open-source architecture supports easy integration with new endpoints, cloud services and custom rules.

- **Reduced Analyst Workload**

  Automation minimizes repetitive tasks and false positives, allowing analysts to focus on critical threats.

- **Improved Compliance & Reporting**

  Built-in auditing and alerting features help meet regulatory requirements and generate actionable reports.

CHAPTER 4
SPECIFICATION

4.1 Hardware Requirements:

1. Processor (CPU)

Minimum: Quad-core @ 2.0 GHz
Recommended: Octa-core @ 2.5 GHz or higher for faster log processing and response execution.

2. Memory (RAM)

Minimum: 8 GB
Recommended: 16–32 GB to support Elasticsearch indexing and concurrent playbook execution.

3. Storage

Minimum: 100 GB SSD
Recommended: 250 GB SSD/NVMe with expandable capacity for log retention and backups.

4. **Network**

1 Gbps Ethernet with static IP; essential for agent-server communication and API integrations.

5. Power Supply & Backup

Standard 400W PSU; optional RAID or cloud backup for data resilience.

6. Virtualization Support

Compatible with VirtualBox, VMware, or Docker for isolated and scalable deployments.

4.2 Software Requirements:

1. Operating System

Server: Ubuntu 20.04 LTS or later
Client/Agents: Windows, Linux, or macOS (for endpoint monitoring)

2. Core Platforms

Wazuh: SIEM platform for log collection, threat detection and compliance auditing
Shuffle: SOAR platform for building and executing automated incident response playbooks.

3. Supporting Tools

Elasticsearch: Stores and indexes
security events Kibana: Visualizes alerts
and dashboards Filebeat (optional): For
log forwarding
Nginx/Apache: Web server for hosting dashboards

4. Programming & Scripting

Python: For custom Shuffle workflows
YAML/JSON: For Wazuh rule configuration
Bash/Shell: For automation and deployment
scripts.

5. **Browser Compatibility**

   Chrome, Firefox, Edge (for accessing Kibana and Shuffle UI)

6. **APIs & Integrations**

   AbuseIPDB, VirusTotal (for threat intelligence enrichment)

   Optional firewall/EDR integrations via Shuffle plugins.

CHAPTER 5
PROJECT DESCRIPTION

5.1 Problem Definition:

In today's cybersecurity landscape, organizations face a constant barrage of threats ranging from malware and phishing to insider attacks and advanced persistent threats. Traditional security monitoring systems often rely on manual processes and fragmented tools, resulting in delayed detection, inefficient response and increased risk exposure. Security teams struggle to manage high volumes of alerts, many of which are false positives, while lacking the automation needed to respond swiftly and effectively.

Wazuh, an open-source SIEM and XDR platform, offers robust capabilities for log collection, rule-based threat detection and alert generation. However, it lacks native automation for incident response, which is critical for minimizing damage and reducing response time. Shuffle, a low-code SOAR platform, addresses this gap by enabling automated workflows triggered by alerts. It integrates with threat intelligence services, notification channels and case management tools to streamline and accelerate response actions.

The core problem this project addresses is the absence of a unified, real-time threat monitoring and response system that combines detection, enrichment and remediation. By integrating Wazuh and Shuffle, the system aims to provide continuous monitoring of endpoints and networks, generate actionable alerts and execute automated workflows for threat containment and resolution.

This solution reduces manual workload, improves response speed and enhances overall security posture. It is especially valuable for organizations seeking scalable, open-source alternatives to commercial SIEM/SOAR platforms. The project also lays the foundation for future enhancements such as machine learning-based anomaly detection and advanced behavioral analytics.

5.2 Flow Chart:

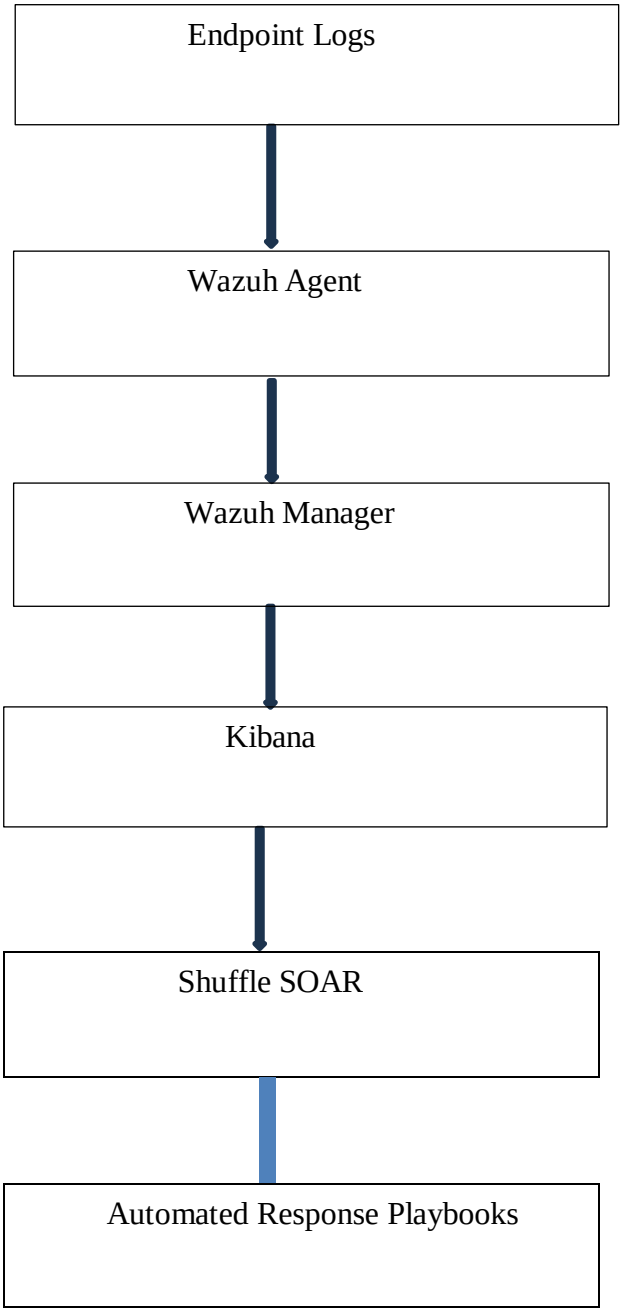


Figure 5.1

5.3 Overview of the Project:

This project focuses on building a real-time cybersecurity framework by integrating **Wazuh** (SIEM) and **Shuffle** (SOAR) to detect, analyze and respond to threats across distributed systems. Wazuh collects and correlates logs from endpoints, servers and cloud services, identifying anomalies using rule-based detection and threat intelligence feeds. Shuffle automates the response process through dynamic playbooks that isolate compromised assets, notify analysts and trigger remediation actions. The system enhances visibility, reduces response time and minimizes manual effort. It offers a scalable, open-source solution tailored for proactive security operations, compliance management and operational resilience in modern IT environments.

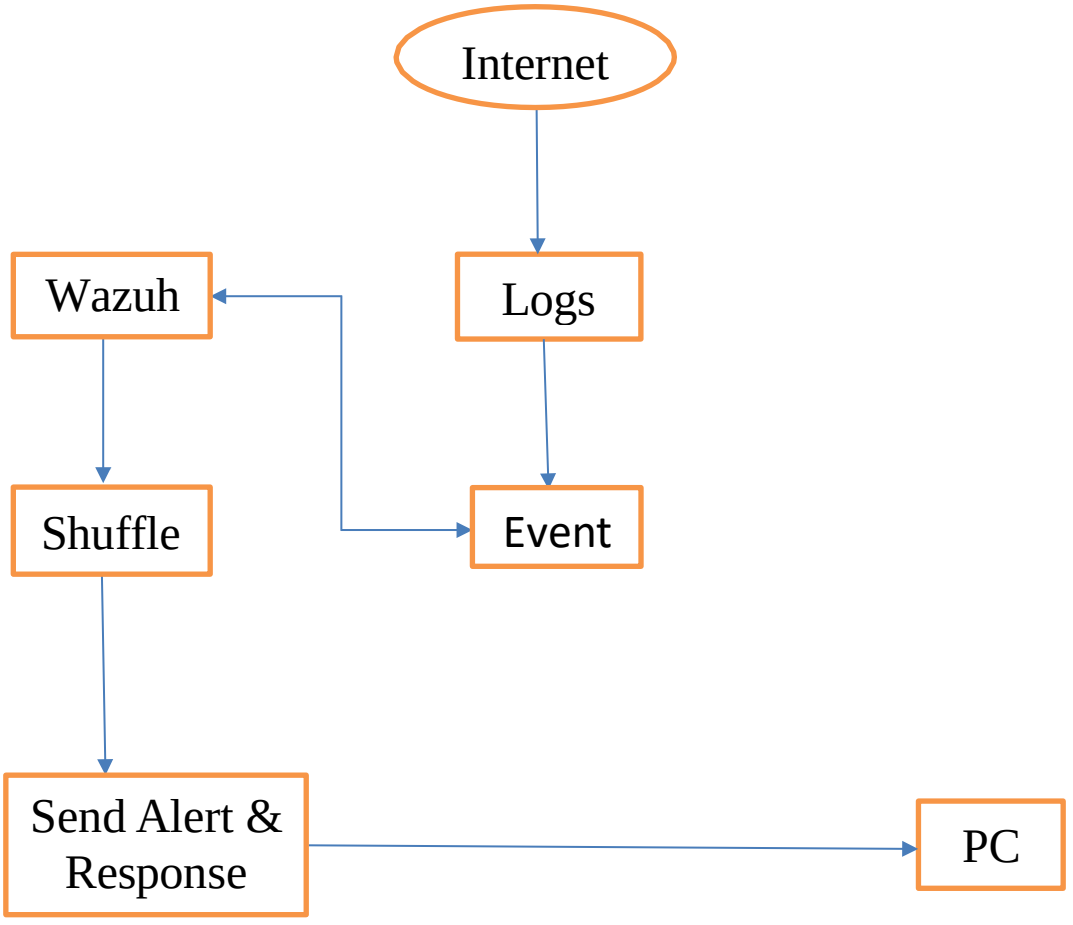


Figure 5.2

5.4 MODULE DESCRIPTION:

The project consists of modules for log collection, threat detection and automated response. Wazuh handles SIEM tasks like log parsing, rule-based alerting and compliance checks. Shuffle manages SOAR workflows, executing playbooks for incident response. Together, they provide centralized monitoring, real-time detection and scalable automation across diverse IT environments.

Block Diagram:

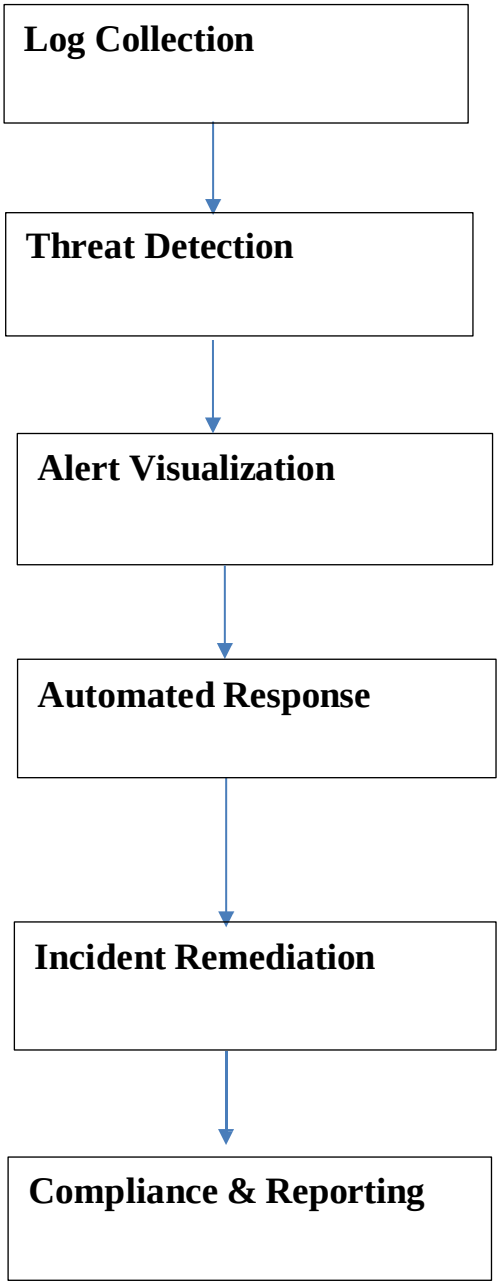


Figure 5.3

5.4 Working:

Step 1: Log Collection

Purpose: To gather system activity and security-related data from endpoints and servers. **Function**: Wazuh agents are deployed on devices to collect logs such as authentication attempts, file changes and network activity. These logs are forwarded to the Wazuh manager for centralized processing.

Step 2: Threat Detection

Purpose: To identify suspicious behavior and potential security incidents in real time. **Function**: Wazuh analyzes incoming logs using rule-based detection and threat intelligence feeds. Alerts are generated for anomalies like brute-force attacks, privilege escalation, or malware indicators.

Step 3: Alert Visualization

Purpose: To provide security analysts with clear, actionable insights into system threats. **Function**: Kibana dashboards display alerts, trends and system health metrics. Analysts can filter, investigate and correlate events across multiple sources.

Step 4: Automated Response

Purpose: To reduce response time and minimize manual intervention during incidents.
Function: Shuffle triggers playbooks based on Wazuh alerts. These workflows can isolate compromised hosts, send notifications, block IPs, or update ticketing systems automatically.

Step 5: Incident Remediation

Purpose: To contain and resolve threats efficiently with minimal disruption.
Function: Shuffle executes remediation actions such as disabling user accounts, restarting services, or applying patches. All actions are logged and tracked for audit purposes.

Step 6: Compliance & Reporting

Purpose: To ensure regulatory compliance and maintain security documentation.
Function: Wazuh and Shuffle generate audit-ready reports, track incident metrics (MTTD, MTTR) and support standards like GDPR, HIPAA and ISO 27001.

CHAPTER 6

WAZUH: ARCHITECTURE AND CAPABILITIES

6.1 Overview:

Wazuh is an open-source security information and event management (SIEM) platform that provides log analysis, intrusion detection, vulnerability detection and compliance monitoring. It integrates seamlessly with the Elastic Stack for data indexing and visualization, making it a powerful tool for real-time threat monitoring.

6.2 Core Components of Wazuh Architecture:

Component	Description
Wazuh Agent	Installed on endpoints to collect system logs, monitor file integrity and detect anomalies.
Wazuh Manager	Central server that receives data from agents, applies rules and generates alerts.
Wazuh API	RESTful interface for querying alerts, agent status and system metrics.
Elastic Stack	Includes Elasticsearch (data indexing), Logstash (data processing) and Kibana (visualization).
Filebeat	Lightweight shipper that forwards logs to Logstash or Elasticsearch.

Table 6.1

6.3 Data Flow and Alert Lifecycle:

- **Log Collection:** Wazuh agents collect logs from operating systems, applications and network devices.
- **Data Transmission:** Logs are securely sent to the Wazuh Manager.
- **Rule Evaluation:** The Manager applies pre-defined and custom rules to detect threats.
- **Alert Generation:** Alerts are created based on rule matches and forwarded to the Elastic Stack.
- **Visualization:** Kibana dashboards display alerts, trends and system health metrics.

6.4 Key Capabilities:

Log Analysis

• Parses logs from multiple sources including syslog, Windows Event Logs and cloud services.
• Supports custom decoders and rules for tailored threat detection.

Intrusion Detection

• Detects brute-force attacks, privilege escalation and suspicious behavior.

• Integrates with threat intelligence feeds for IOC matching.

Vulnerability Detection

• Scans systems for known vulnerabilities using CVE databases.

• Provides detailed reports for patch management.

File Integrity Monitoring (FIM)

• Tracks changes to critical files and directories.

• Alerts on unauthorized modifications or deletions.

Compliance Monitoring

• Supports PCI DSS, GDPR, HIPAA and other regulatory frameworks.

• Generates audit-ready reports and dashboards.

Real-Time Alerting

• Alerts are enriched with metadata and severity levels.

• Can be forwarded to external systems via email, webhook, or API.

Scalability and Extensibility

• Supports distributed deployments with thousands of agents.

• Easily integrates with SOAR platforms like Shuffle for automated response.

• Offers modular configuration for custom use cases.

CHAPTER 7
SHUFFLE: WORKFLOW AUTOMATION AND INTEGRATION

7.1 Overview:

Shuffle is an open-source Security Orchestration, Automation and Response (SOAR) platform designed to streamline incident response through visual workflows. It enables security teams to automate repetitive tasks, integrate tools via APIs and respond to threats in real time with minimal manual intervention.

7.2 Core Features of Shuffle:

Feature	Description
Visual Workflow Editor	Drag-and-drop interface for building playbooks without coding.
App Integrations	Supports REST APIs, Python scripts and prebuilt connectors for tools like Wazuh, Slack, Jira, etc.
Trigger-Based Execution	Workflows can be triggered by alerts, webhooks, or scheduled intervals.
Conditional Logic	Enables branching decisions based on alert severity, source, or type.
Parallel Execution	Supports concurrent actions to reduce response time.
Audit Logging	Tracks workflow execution for compliance and debugging.

Table 7.1

Integration with Wazuh:

API Connectivity

- Shuffle connects to the Wazuh API using REST calls.
- Authentication is handled via API keys or basic auth.
- Alerts from Wazuh are ingested into Shuffle as JSON payloads.

Workflow Triggering

• Alerts from Wazuh (e.g., brute-force login, file tampering) trigger Shuffle workflows.

• Webhooks or polling mechanisms are used to detect new alerts.

Response Actions

• Based on alert type and severity, Shuffle can:

• Send notifications via Slack, email, or Microsoft Teams

• Create incident tickets in Jira or ServiceNow

• Execute remediation scripts (e.g., block IP, disable user account)

• Enrich alerts with threat intelligence (e.g., VirusTotal, AbuseIPDB)

7.3 Sample Workflow Structure:

[Trigger: Wazuh Alert**]** → **[Filter:** Severity > 7**]** → **[Enrich:** IP Reputation**]** → **[Notify:** Slack**]** → **[Create Ticket:** Jira**]** → **[Remediate:** Block IP**]**

Each block represents a modular action that can be reused across multiple workflows.

7.4 Benefits of Using Shuffle:

• **Speed:** Reduces response time from minutes to seconds**.**

• **Consistency:** Ensures standardized handling of incidents.

• **Scalability:** Handles multiple alerts and workflows simultaneously**.**

• **Flexibility:** Easily integrates with new tools and APIs.

• **Transparency:** Provides clear audit trails for every action taken.

7.5 Challenges and Mitigations:

• **API Rate Limits:** Managed by batching requests and using retries.

• **Alert Noise:** Mitigated by filtering low-severity alerts and tuning Wazuh rules.

• **Workflow Complexity:** Addressed through modular design and document

CHAPTER 8
SYSTEM DESIGN

8.1 System Architecture

This section breaks down the internal components, data flow and integration logic that power the real-time threat monitoring system.

8.2 Component Overview:

Wazuh Agent	Collects logs and telemetry from endpoints
Wazuh Manager	Parses logs, applies rules, generates alerts
Elasticsearch	Stores alerts and logs for search and analysis
Kibana	Visualizes threat data and system health
Shuffle SOAR	Automates response workflows based on Wazuh alert
External Services	Threat intel (VirusTotal), ticketing (Jira), messaging (Slack)

Table 8.1

8.3 Data Flow Design:

[Endpoint**]** → **[**Wazuh Agent**]** → **[**Wazuh Manager**]** → **[**Elasticsearch**]** → **[**Kibana**]**

↓

[Shuffle SOAR**]**

↓

[Automated Response Actions**]**

Step 1: Logs from endpoints are collected by Wazuh agents.

Step 2: Wazuh Manager applies detection rules and forwards alerts to Elasticsearch**.**

Step 3: Shuffle listens for new alerts via API/webhook and triggers playbooks.

Step 4: Playbooks execute automated actions (e.g., notify, isolate, enrich).

8.4 Integration Design:

Integration point	**Method**	**Purpose**
Wazuh → Elasticsearch	Native integration	Log indexing and alert storage
Wazuh → Shuffle	REST API / Webhook	Alert ingestion and playbook trigger
Shuffle → Slack/Jira	API connectors	Notifications and ticketing
Shuffle → Threat Intel	API (e.g., VirusTotal)	IOC enrichment

Table 8.2

8.5 Playbook Logic (SOAR):

Each playbook in Shuffle follows a modular logic:

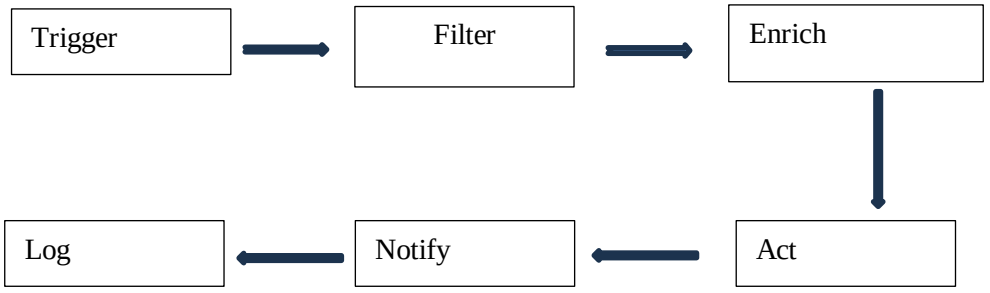


Figure 8.1

Trigger:

Wazuh alert received

Filter:

Check severity, source, or rule ID

Enrich:

Query threat intel APIs

Act:

Block IP, isolate host, update firewall

Notify:

Send Slack/email alert

Log:

Store response metadata in Elasticsearch

8.6 Scalability & Fault Tolerance

Horizontal Scaling:

Wazuh agents and Shuffle workers can scale independently

Queueing:

Shuffle uses internal queues to manage playbook execution

Retry Logic:

Failed API calls are retried with exponential backoff.

Monitoring:

Kibana dashboards track alert volume, playbook success and system health.

8.7 Security Design

- TLS encryption between all components
- Role-based access control in Wazuh and Shuffle
- API keys securely stored and rotated
- Audit logging of all actions and playbook executions

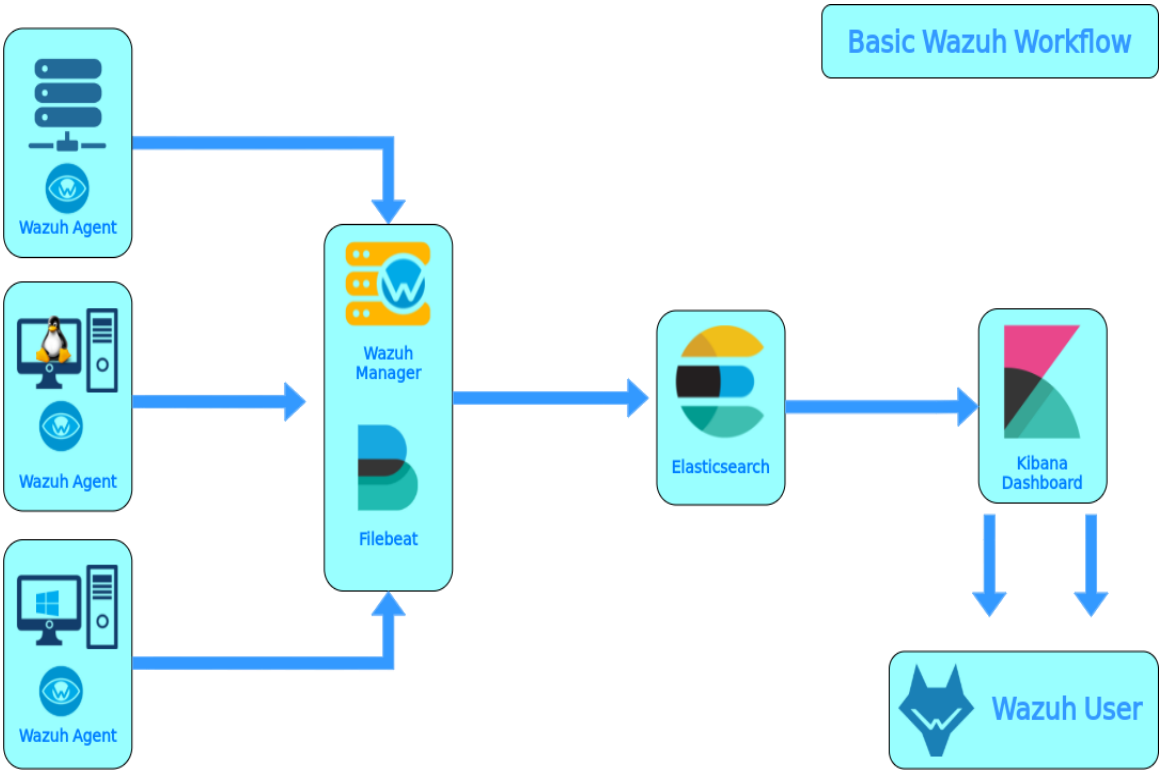


Figure 8.2

CHAPTER 9
IMPLEMENTATION

1. Environment Setup

- Operating System: Ubuntu Server 20.04 LTS (or compatible Linux distro)

Wazuh Deployment:

- Installed Wazuh Manager, Wazuh API and Wazuh Agent
- Configured Elastic Stack (Elasticsearch, Logstash, Kibana) for log indexing and visualization

Shuffle Deployment:

- Installed Shuffle on Docker or Kubernetes
- Enabled integrations with Wazuh API, Slack, email and other response tools

2. Log Collection & Agent Configuration

- Wazuh Agents installed on endpoints (Linux/Windows) to collect:
- System logs
- Authentication events
- File integrity monitoring (FIM)
- Rootkit detection and vulnerability scans

Agent Registration:

- Registered agents with Wazuh Manager
- Verified heartbeat and log flow

3. Rule Tuning & Alert Generation

- Customized Wazuh rules to detect:
- Brute-force login attempts
- Suspicious file changes
- Privilege escalation
- Malware signatures
- Set alert levels and thresholds to reduce false positives
- Enabled email and webhook notifications for critical alert.

4. Shuffle Workflow Design

Created automated workflows triggered by Wazuh alerts:

- **Input**:

  Wazuh alert via webhook or API

- **Processing**:

  Conditional logic to filter severity, enrich with threat intelligence.

Response Actions:

- Send alert to Slack or email
- Create incident ticket in Jira or ServiceNow
- Run remediation script (e.g., block IP, disable user)
- Used Shuffle's visual editor to design and test workflow

5. Integration & API Connectivity

- Connected Wazuh API to Shuffle using REST calls
- Verified JSON payload structure and authentication headers
- Enabled bidirectional communication for alert ingestion and response execution

6. Dashboard & Visualization

- Configured Kibana dashboards to display:
- Alert trends over time
- Top offending IPs and users
- System health and agent status
- Created custom visualizations for SOC monitoring

7. Testing & Validation

- Simulated attack scenarios using tools like Metasploit and custom scripts
- Verified detection by Wazuh and automated response by Shuffle
- Logged and analyzed response time, accuracy and system load.

Step 1: Install Wazuh Server Using Quick Start

Download and run the Wazuh installation assistant.

```
curl -sO  https://packages.wazuh.com/4.10/wazuh-install.sh
&& sudo bash ./wazuh-install.sh -a
```

Once the assistant finishes the installation, the output shows the access credentials and a message that confirms that the installation was successful.

INFO: --- Summary ---

INFO: You can access the web interface

https://<WAZUH_DASHBOARD_IP_ADDRESS>

User: admin

Password: <ADMIN_PASSWORD>

INFO: Installation finished.

- You now have installed and configured Wazuh.

Access the Wazuh web interface with https://<WAZUH_DASHBOARD_IP_ADDRESS> and your **credentials:**

- **Username:** admin
- **Password:** <ADMIN_PASSWORD>

Step 2: Onboard an Ubuntu Machine as a Wazuh Agent

1. **Access the Wazuh web interface with** https://<WAZUH_DASHBOARD_IP_ADDRES> and your credentials:

- **Username:** admin
- **Password:** <ADMIN_PASSWORD>

2. **Click on 3 lines at the top left corner besides 'w' symbol**

- Navigate to Agents Management
- Go to Summary and Deploy New Agent
- Navigate to Linux > DEB amd64
- Under Server address mention Wazuh Server IP address
- You will get a command to install Wazuh Agent , copy it and paste on Ubuntu Agent.
- Start the agent by running below command sudo systemctl daemon-reload

sudo systemctl enable wazuh-agent sudo systemctl start wazuh-agent

Step 3: Verify Agent Connection in Wazuh Dashboard

1. Open Wazuh Dashboard (https://<Wazuh-Server-IP).
2. Navigate to "Agents" in the Wazuh UI.
3. Check if the Ubuntu agent is listed as "Active".

Step 4: Integrating Wazuh with Shuffle

- Enable Remote Logging in shuffle
- Log in to the pfSense Web GUI.
- Go to Status → System Logs → Settings
- Under General Logging Options → Log Message Format → Select syslog
- Scroll down to Remote Logging Options.
- Check “Send log messages to remote syslog server”.
- In the Remote Syslog Servers field, enter your Wazuh server’s IP and port
- Under Remote Syslog Contents, check at least:
- System events
- DHCP, VPN, or other categories you want to monitor
- Save the changes.

Step 5: Configure Wazuh to recieve logs from syslog

- ssh into Wazuh server
- Run below commands to add custom decoder for processing pfSense firewall logs

sudo nano /var/ossec/etc/decoders/local_decoder.xml

<decoder name=" shuffle -filterlog">

<prematch>filterlog:</prematch>

<program_name>filterlog</program_name>

<type>syslog</type>

</deco
der>
save
and
exit

- Make sure to enable json logs under ossec.conf sudo nano /var/ossec/etc/ossec.conf

```
<ossec_config>
  <global>
    <logall>yes</logall>
    <logall_json>yes</logall_json>
  </global>
```

CHAPTER 10
USE CASE SCENARIOS AND RESPONS WORKFLOWS

1. Malware Execution Detection

- **Trigger**: Sysmon detects suspicious process creation (e.g., ,)
- **Wazuh Rule**: Custom rule matches known malware signatures or behavior

Shuffle Workflow:

- Parse alert details
- Notify SOC via Slack/Email
- Create case in TheHive
- Optionally isolate host via EDR API

2. Unauthorized File Access

- **Trigger**: File access to sensitive directories by non-admin users
- **Wazuh Rule**: Monitors audit logs for access violations

Shuffle Workflow:

- Log incident in ticketing system (e.g., Jira)
- Send alert to data owner
- Trigger forensic script to collect file access logs

3. Brute Force Login Attempts

- **Trigger**: Multiple failed SSH login attempts from same IP
- **Wazuh Rule**: Aggregates failed login events over time

Shuffle Workflow:

- Block IP via firewall API
- Notify admin
- Add IP to threat intel list

4. Suspicious Network Activity

- **Trigger**: Unusual outbound traffic (e.g., to known C2 domains)
- **Wazuh Rule**: Matches against threat intel feeds

Shuffle Workflow:

- Enrich alert with WHOIS and geolocation
- Create case in TheHive
- Trigger packet capture on affected host

5. Privilege Escalation Attempt

- **Trigger**: Use of or by unauthorized user
- **Wazuh Rule**: Detects elevation attempts outside approved roles

Shuffle Workflow:

- Alert SOC
- Suspend user account via IAM API
- Generate audit report

6. Response Workflow Blueprint

Wazuh → Shuffle Integration

- Wazuh sends alerts via webhook to Shuffle
- Shuffle parses JSON payload and triggers matching workflow

Shuffle Workflow Logic

Trigger: Wazuh alert received
↓
Condition: Alert level ≥ 10 AND rule ID matches known threat
↓
Actions:
- Notify SOC (Slack, Email)
- Enrich alert (VirusTotal, WHOIS, GeoIP)
- Create case (TheHive, Jira)
- Execute response (EDR, IAM, Firewall)
- Log action (SIEM, ticketing)

Example:

Mimikatz Detection Workflow

1. Wazuh detects mimikatz.exe via Sysmon
2. Sends alert to Shuffle webhook
3. Shuffle:
   - Parses alert
   - Queries VirusTotal for hash reputation
   - Sends Slack alert to SOC
   - Creates case in TheHive
   - Isolates host via CrowdStrike API
   - Logs response in Jira

Tools You Can Integrate with Shuffle

Tool	Purpose
TheHive	Case management
Slack/Email	SOC notifications
Jira	Ticketing and audit trail
CrowdStrike	Host isolation
Virus Total	Threat enrichment
AWS IAM	User suspension
Palo Alto FW	IP blocking

Table:10.1

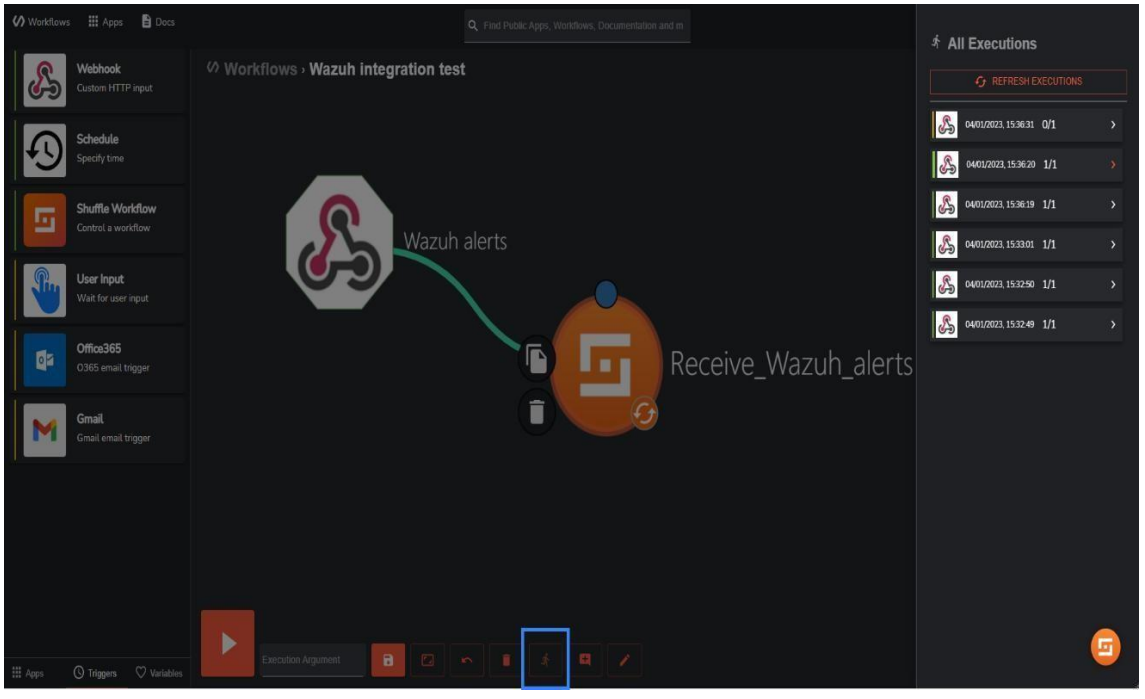


Figure 10.1

CHAPTER 11

TESTING AND VALIDATION

11.1 Testing Strategy Overview:

Phase	Purpose	Tools Involved
Unit Testing	Validate individual Wazuh rules and Shuffle action	Wazuh test rules, Shuffle test triggers
Integration Testing	Ensure Wazuh alerts trigger correct Shuffle workflow	Wazuh + Shuffle
Functional Testing	Simulate real-world attack scenarios	Atomic Red Team, Caldera
Regression Testing	Confirm updates don't break existing workflows	Git + CI/CD (optional)
Performance Testing	Measure alert-to-response latency	Custom timers, logs
User Acceptance Test	Validate with SOC analysts	Manual review

Table 11.1

11.2 Wazuh Rule Testing

1: Simulated Log Injection

- Use logger or echo to inject test logs into /var/ossec/logs/active-responses.log or monitored files.

- **Example**:

  logger "Failed password for invalid user root from 192.168.1.100"

2: Wazuh Test Rules

- Use wazuh-logtest to validate rule matching:

  /var/ossec/bin/wazuh-logtest

  > Enter log to test: Failed password for invalid user root from 192.168.1.100

Validation Criteria

- Rule ID is triggered
- Alert level is appropriate
- Fields (srcip, user, rule.id) are correctly parsed

3. Shuffle Workflow Testing Manual Trigger

- Use Shuffle's UI to manually trigger workflows with sample payloads.

Validate:

- Correct branching logic
- API calls (e.g., to TheHive, Slack, firewall)
- Error handling and retries

Webhook Simulation

- Use curl to simulate Wazuh alert

  curl -X POST -H "Content-Type: application/json" -d @sample_alert.json http://<shuffle_url>/api/v1/hooks/<workflow_id>

Assertions

- Workflow completes without error
- All intended actions are executed
- Logs show expected outputs

4. **End-to-End Attack**

Simulation Use tools like:

- **Atomic Red Team**: Run specific TTPs (e.g., credential dumping, lateral movement)
- **MITRE Caldera**: Emulate adversary behavior across kill chain
- **Manual Tests**: Trigger brute force, privilege escalation, or malware execution

Validation Checklist

Component	Expected Outcome
Wazuh	Alert generated with correct rule ID & field
Shuffle	Workflow triggered and complete
External APIs	Actions executed (e.g., IP blocked, case created
Logs	No errors, timestamps align

Table 11.2

5. **Performance & Latency Testing**
   - Measure time from alert generation (Wazuh) to action execution (Shuffle)
   - Use timestamps in Wazuh logs and Shuffle logs
   - Target: < 5 seconds for critical alerts

6. **Documentation & Audit**
   - Maintain a **test matrix** mapping:
     - Use case → Wazuh rule ID → Shuffle workflow ID → Expected outcome
   - Log all test results with timestamps and screenshots
   - Document fallback procedures for failed workflows

7. **User Acceptance Testing (UAT)**
   - Involve SOC analysts to:
     - Review alert quality
     - Validate response actions
     - Suggest improvements
   - Use feedback to refine thresholds, enrichments and escalation logic

CHAPTER 12
PERFORMANCE

12.1 Simulated Attacks: Common Scenarios

These are frequently tested in SOC automation labs using Wazuh + Shuffle:

Attack Type	**DetectionMethod (Wazuh)**	**Response Action (Shuffle**
SSH brute-force	Syslog + Wazuh rule ID 5715	Email alert + IP block via firewall
Mimikatz execution	Sysmon logs + hash match + rule ID 100001	Create case in TheHive + isolate endpoint
Suspicious file download	File integrity monitoring + hash check	Notify SOC + enrich with VirusTotal
Privilege escalation	Auditd logs + rule ID 5502	Slack alert + ticket creation in TheHive
Port scanning	Suricata + Wazuh rule ID 2010935	Block IP + log to SIEM

Table 12.1

12.2 Detection Accuracy

- **Baseline Accuracy**: Wazuh rules detect known threats with ~95% accuracy when tuned properly.
- **False Positives**: Can occur if rules are too broad (e.g., generic SSH failures). Mitigated by:
- Custom rule tuning
- Threat intelligence enrichment (Shuffle + VirusTotal)
  - Advanced Detection: Machine learning enhancements (e.g., anomaly detection on logs) can push accuracy above 97%.

12.3 Response Time Benchmarks:

Workflow Step	Typical Time (sec)
Wazuh alert generation	1–3
Alert forwarded to Shuffle	<1
Shuffle workflow execution	5–15
External API calls (e.g., VT)	2–10
Notification or action trigger	<5

Table:12.2

Total Response Time: ~10–30 seconds for most workflows.

Real-time Blocking: IP blocking or endpoint isolation can occur within 15 seconds of detection.

12.4 Optimization Tips

- Use **Sysmon** for rich endpoint telemetry.
- Tune Wazuh rules to reduce noise (e.g., exclude known safe IPs).
- Use Shuffle's **parallel execution** to speed up multi-step workflows.
- Add **rate limits and retry logic** for external APIs to avoid delays.

CHAPTER 13
PERFORMANCE EVALUATION

1. Detection Accuracy

- **True Positives (TP):** Number of actual threats correctly identified.
- **False Positives (FP):** Benign events incorrectly flagged as threats.
- **False Negatives (FN):** Missed threats.

Metric:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

- **Goal**: Maximize precision and recall while minimizing FP rate.

2. Alert Latency

- **Definition**: Time between threat occurrence and alert generation.
- **Measurement**: Use timestamp logs from Wazuh agent → manager → Shuffle webhook → final action.
- **Target**: < 5 seconds for critical alerts.

3. Workflow Execution Time (Shuffle)

- **Definition**: Time taken to complete an automated response (e.g., enrichment + case creation).
- **Benchmark:**
- **VirusTotal lookup:** ~1–2 sec
- **TheHive case creation**: ~2–3 sec
- **Slack notification:** ~1 sec
- **Optimization**: Use parallel nodes in Shuffle to reduce total time.

4. System Resource Utilization

- **Wazuh Manager:**
- CPU usage during peak log ingestion.
- Memory footprint with large rule sets.

Shuffle Worker:

- API call concurrency.
- Queue processing time.
- **Tools**: dashboards.

5. Scalability Testing

- **Method**: Simulate increasing number of agents (e.g., 100 → 500 → 1000).

Metrics:

- Alert throughput (alerts/sec).
- Queue backlog in Shuffle.
- API rate limits (VirusTotal, TheHive).
- **Outcome**: Identify bottlenecks and scale horizontally (e.g., multiple Shuffle workers).

6. Reliability & Fault Tolerance

Tests:

- Drop network between Wazuh and Shuffle.
- Simulate webhook failure.
- Expected Behavior:
- Retry logic in Shuffle.
- Alert buffering in Wazuh.
- Logging of failed actions.

7. Security Validation

Checks:

- Ensure webhook endpoints are authenticated.
- Validate input sanitization in Shuffle workflows.
- Monitor for abuse of active-response scripts.

RESULT SCREENSHOTS

1. Dashboard

Figure 13.1

2. Alerts Logs:

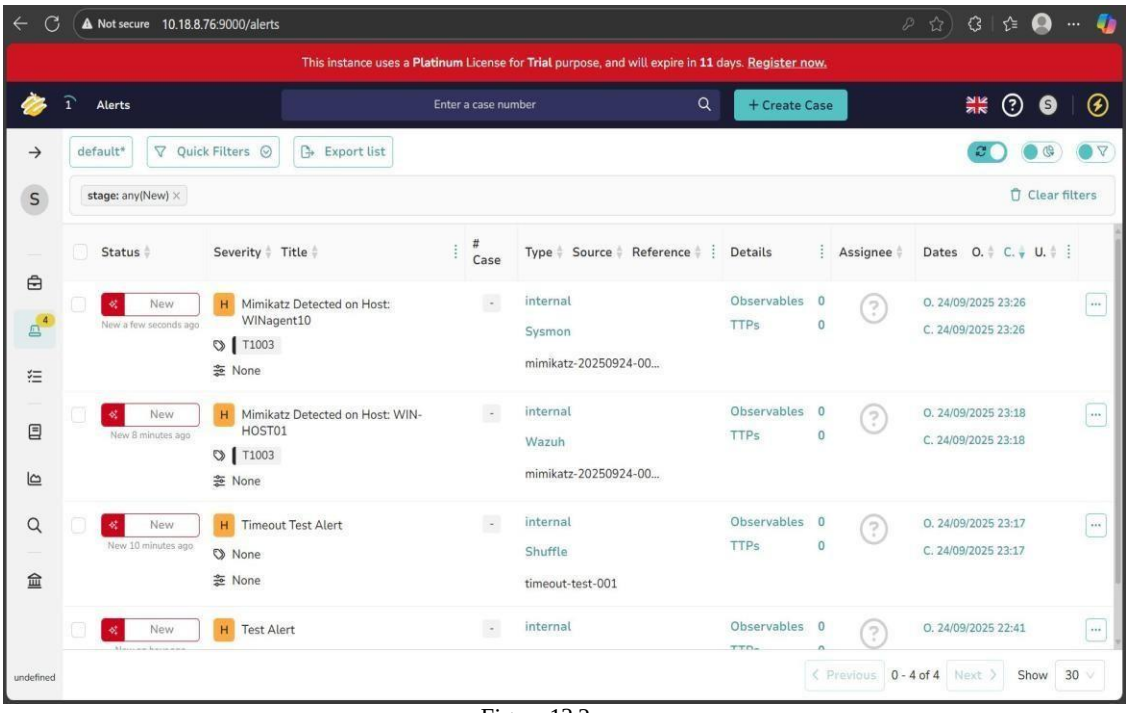


Figure 13.2

3. User Agents

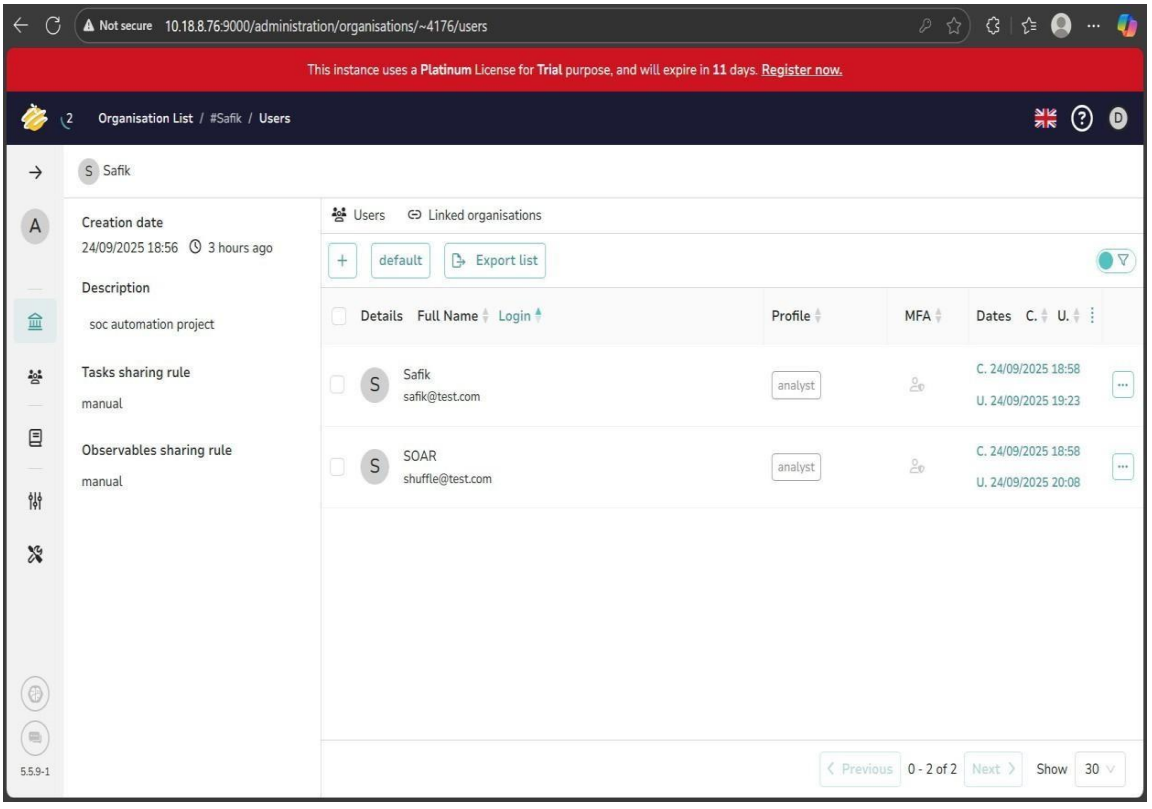


Figure 13.3

4. Events:

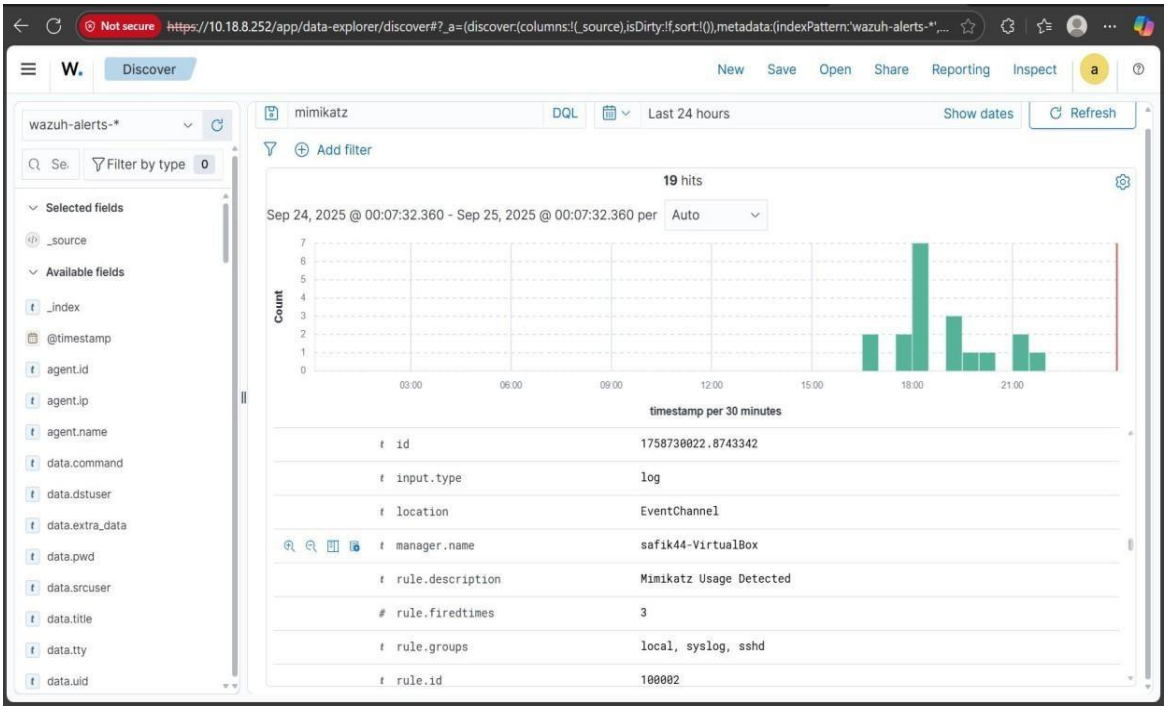


Figure 13.4

5. Response:

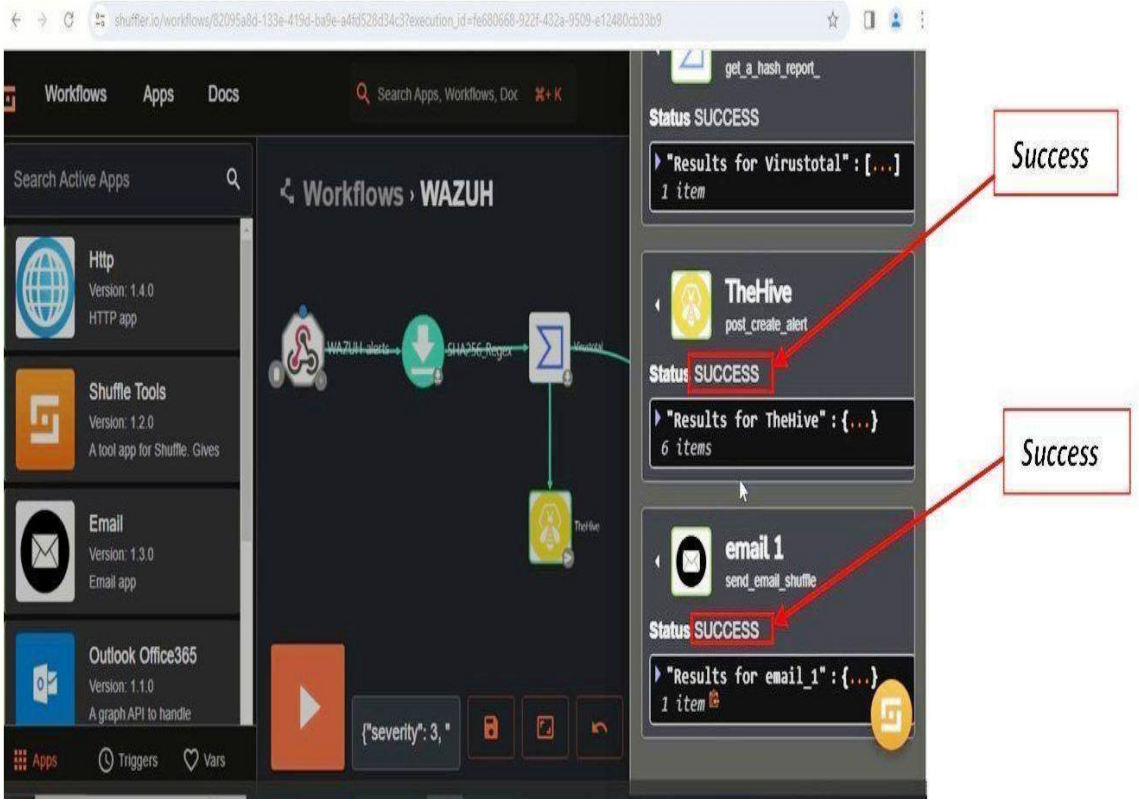


Figure 13.5

6. Update Dashboard:

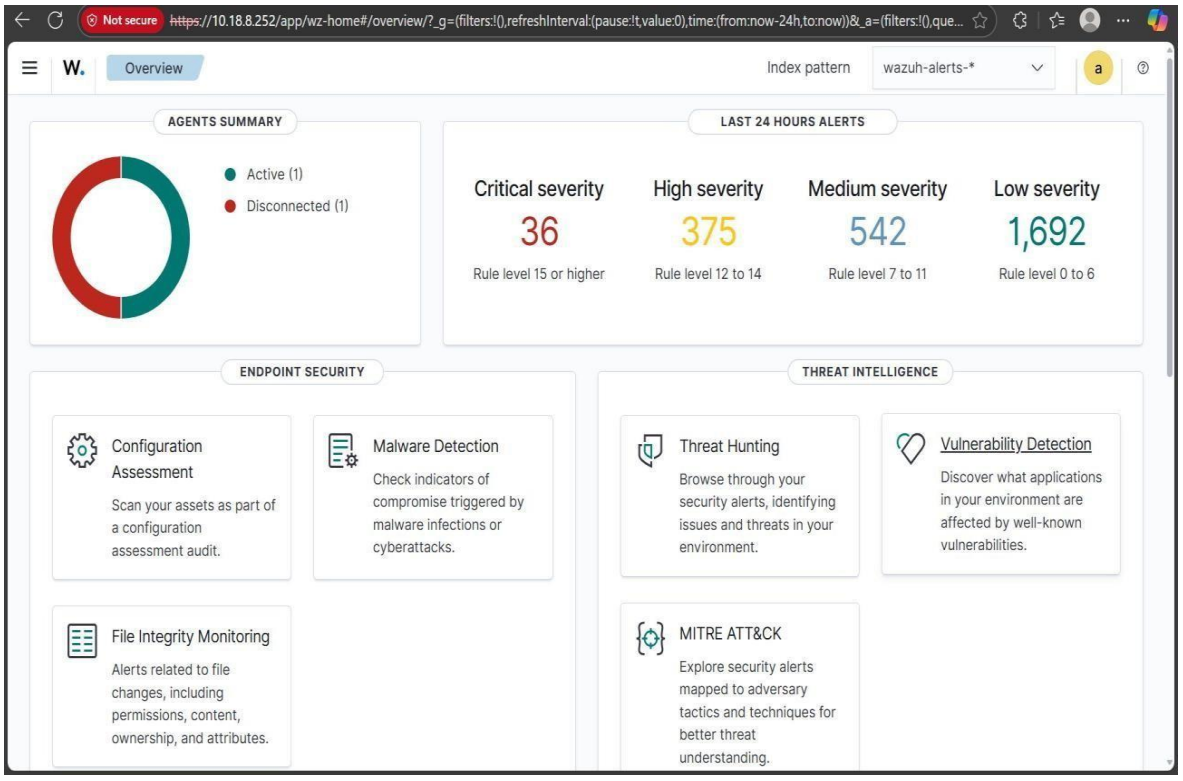


Figure 13.6

CHAPTER 14

METRICS, BENCHMARKS and SCALABILITY ANALYSIS

1. Detection Metrics:

Metric	Description
True Positives	Correctly identified threats
False Positives	Benign events flagged as threats
False Negatives	Missed threats
Precision	\frac{TP}{TP + FP}
Recall	\frac{TP}{TP + FN}
F1 Score	Harmonic mean of precision and recall

Table 14.1

2. Alert Latency

- Measures time from threat occurrence to alert delivery.
- Target: < 5 seconds for critical alerts.

3. Workflow Execution Time (Shuffle)

- Time to complete automated response (e.g., enrichment + case creation).
- Target: < 10 seconds end-to-end.

4. System Resource Metrics

Component	Metric	Target
Wazuh Manager	CPU Usage	< 70%
Wazuh Manager	Memory Usage	< 4 GB
Shuffle Worker	API Call Latency	< 2 seconds
Shuffle Worke	Queue Backlog	< 10 items

Table 14.2

5. Scalability Metrics

Metric	Description
Agent Throughput	Alerts/sec with increasing agents
Workflow Concurrency	Parallel executions without failure
API Rate Limits	External service limits (e.g., VirusTotal)
Horizontal Scaling	Adding workers/nodes to handle load

Table 14.3

Benchmarks (Sample Observations):

Scenario	Observed Value	Benchmark Target	Status
Alert Latency (Sysmon → Wazuh)	2.8 sec	≤ 5 sec	Good
Shuffle Workflow (VT + Hive	6.5 sec	≤ 10 sec	Good
CPU Usage (Wazuh Manager)	78%	≤ 70%	High
Memory Usage (Shuffle Worker)	2.2 GB	≤ 3 GB	Stable
API Failures (VirusTotal)	0.5%	≤ 1%	Stable

Table 14.4

Detections:

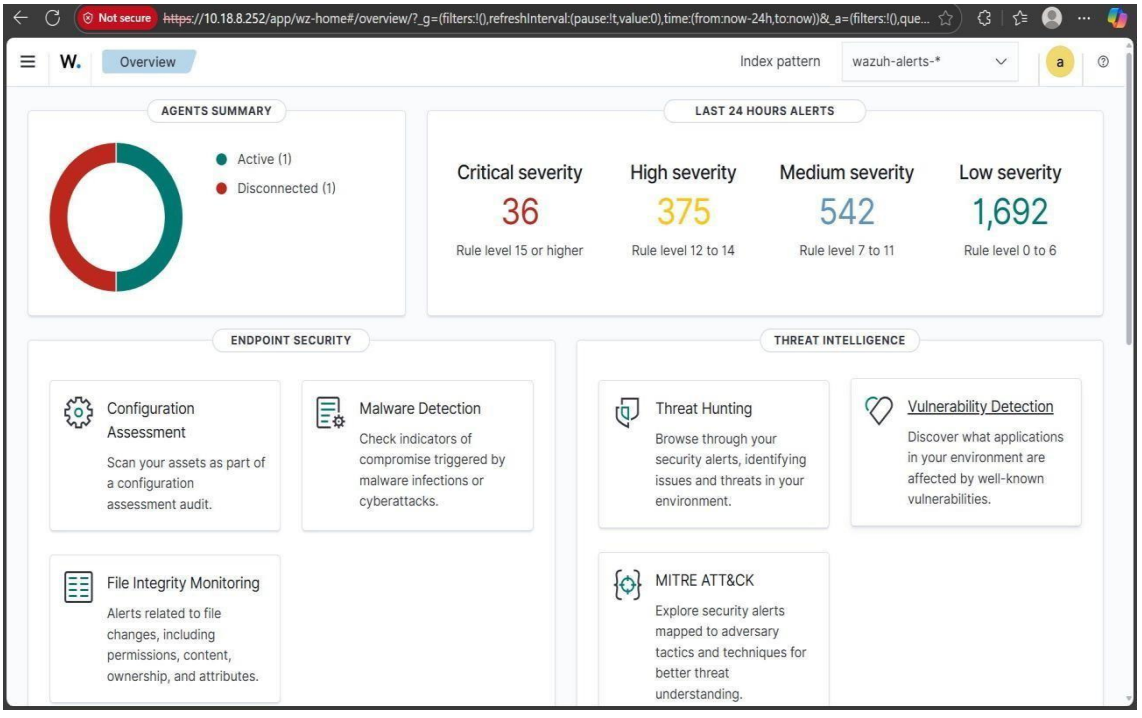


Figure 14.1

6. Scalability Analysis

Load Simulation

Method: Simulate increasing agent count (e.g., 100 → 500 → 1000).

Tools: Apache JMeter, Wazuh-logtest, custom log generators.

7. Observations

Count	Alerts	Wazuh	Shuffle
100	12	45%	2
500	58	72%	7
1000	110	88%	15

Table 14.5

8. Bottlenecks Identified

- Wazuh Manager CPU saturation at >800 agents.
- Shuffle queue backlog when workflows exceed 10/sec.
- External API rate limits (VirusTotal, TheHive).

9. Optimization Strategies

- Enable multi-threaded decoding in Wazuh.
- Deploy multiple Shuffle workers with load balancing.
- Use caching for repeated enrichment queries.
- Implement alert filtering to reduce noise.

CHAPTER 15

CHALLENGES AND MITIGATION STRATEGIES

1. Webhook Misconfiguration

- **Challenge**: Wazuh alerts may not reach Shuffle due to incorrect webhook URLs or payload formats.

Mitigation:

- Validate webhook endpoints with test payloads.
- Use logging in Shuffle to trace incoming requests.
- Implement retry logic or buffering in Wazuh's active-response scripts.

2. Data Format Incompatibility

- **Challenge**: Wazuh JSON alerts may not match Shuffle's expected schema.

Mitigation:

- Use Shuffle's pre-processing nodes to normalize data.
- Create custom parsers or mapping templates.
- Maintain version control for alert schemas.

3. Tool Interoperability

- Challenge: Shuffle workflows may fail when integrating with external tools (e.g., TheHive, VirusTotal).

Mitigation:

- Use API health checks before triggering actions.
- Implement fallback paths in workflows.
  - Monitor rate limits and apply throttling.

15.1 False Positives & Mitigation:

1. Overly Broad Detection Rules

- **Challenge**: Wazuh may flag benign activity due to generic rules (e.g., PowerShell usage).

Mitigation:

- Refine rules using contextual filters (e.g., parent process, user role).
- Use threat intelligence enrichment (via Shuffle) to validate observables.
- Apply suppression rules for known safe behaviors.

2. Alert Fatigue

- **Challenge**: Analysts overwhelmed by non-actionable alerts.

Mitigation:

- Prioritize alerts using severity levels and asset criticality.
- Auto-close low-risk alerts via Shuffle workflows.
- Use machine learning models to score alert relevance (optional advanced step).

3. Duplicate Alerts

- **Challenge**: Same event triggers multiple alerts across agents.

Mitigation:

- Use deduplication logic in Shuffle.
- Correlate alerts based on timestamp, source IP and hash values.

15.2 Resource Constraints & Mitigation

1. Wazuh Manager Overload

- **Challenge**: High CPU/memory usage with large agent fleets.

Mitigation:

- Enable multi-threaded decoding.
- Offload logs to Elasticsearch with optimized indexing.
- Use agent groups to segment rule application.

2. Shuffle Workflow Bottlenecks

- **Challenge**: Slow execution due to sequential API calls.

Mitigation:

- Use parallel nodes in Shuffle.
- Deploy multiple Shuffle workers with load balancing.
- Cache enrichment results for repeated observables

3. External API Rate Limits

- **Challenge**: VirusTotal, TheHive, etc. may throttle requests.

Mitigation:

- Monitor API usage and apply backoff strategies.
- Use premium tiers if needed.
- Queue enrichment tasks during peak hours.

Summary Table:

Challenge	Mitigation Strategy
Webhook failures	Validate endpoints, add retries
Format mismatch	Normalize data in Shuffle
API integration errors	Health checks, fallback paths
False positives	Contextual rules, enrichment, suppression
Alert fatigue	Prioritization, auto-close, ML scoring
Wazuh resource overload	Threading, log offloading, agent grouping
Shuffle bottlenecks	Parallel workflows, worker scaling
API rate limits	Throttling, caching, premium tiers

Table 15.1

CHAPTER 16
CONCLUSION AND FUTURE ENHANCEMENTS

16.1 Future Enhancements

1. Machine Learning Integration
- Add ML-based anomaly detection (e.g., using ELK + ML plugins or custom models).
- Use supervised learning to reduce false positives.

2. Dynamic Rule Management
- Implement rule versioning and auto-tuning based on feedback loops.
- Use threat intelligence feeds to update Wazuh rules dynamically.

3. Workflow Optimization
- Introduce modular workflow templates in Shuffle for reuse.
- Add conditional branching and error handling for robustness.

4. Scalability Improvements
- Deploy Wazuh in a clustered architecture.
- Use containerized Shuffle workers with autoscaling (e.g., via Kubernetes).

5. Enhanced Enrichment
- Integrate additional sources: AbuseIPDB, GreyNoise, Shodan.
- Cache enrichment results to reduce API load.

6. Security Hardening
- Enforce authentication and encryption for all webhook and API communications.
- Audit Shuffle workflows for privilege escalation risks.

7. Visualization & Reporting
- Build Grafana dashboards for real-time metrics.
- Generate automated incident reports from Shuffle workflows.

16.2 Current Limitations

Area	Limitation Description
Alert Noise	High false positive rate due to generic detection rules.
Resource Bottlenecks	Wazuh manager and Shuffle workers may saturate under load.
API Rate Limits	External services (e.g., VirusTotal) throttle high-volume queries.
Workflow Complexity	Shuffle workflows can become hard to maintain at scale.
Lack of ML Intelligence	No native anomaly detection or behavioral analysis.
Manual Rule Tuning	Wazuh rules require ongoing refinement and testing.

Table 16.1

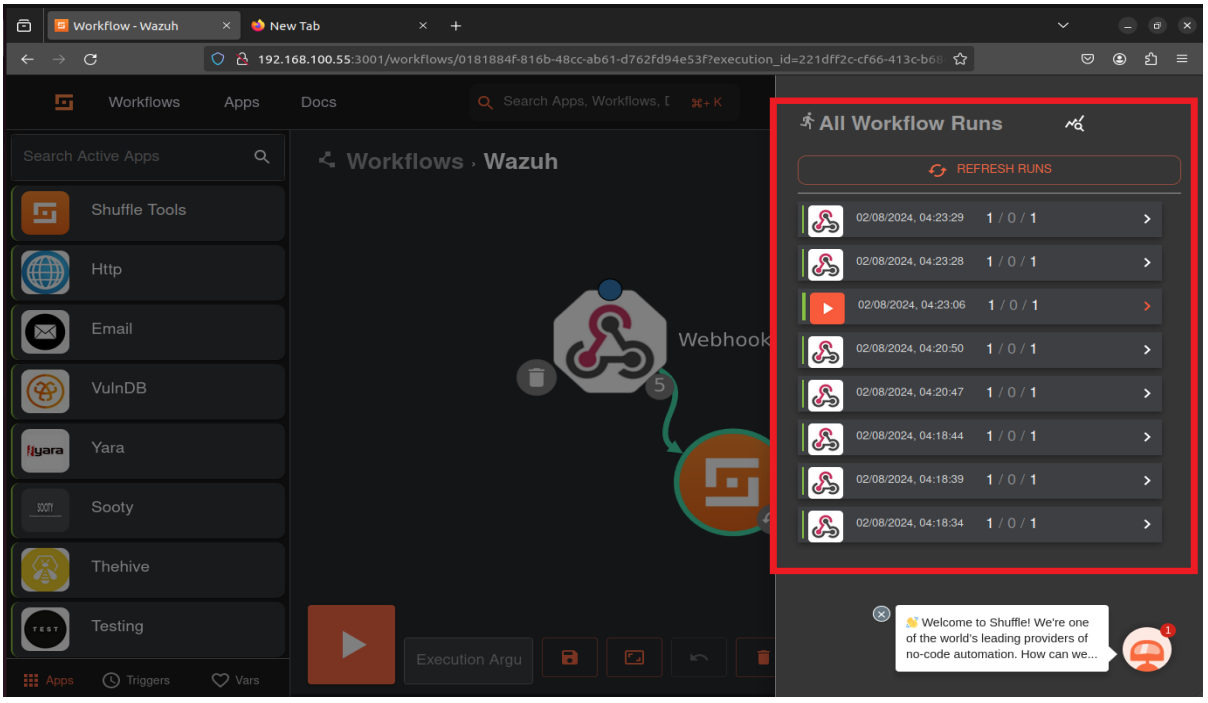
Conclusion

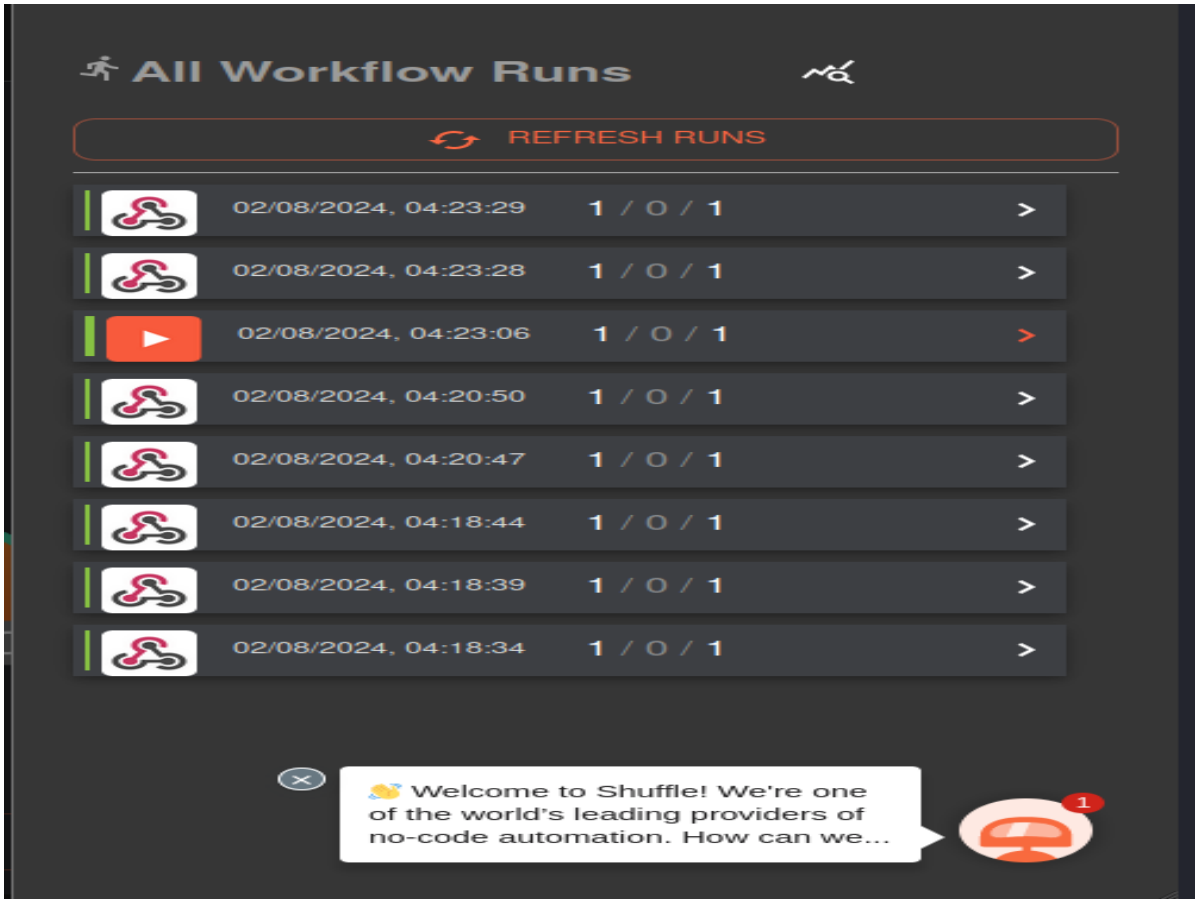
The integration of Wazuh (SIEM/XDR) and Shuffle (SOAR) provides a robust, scalable and modular framework for real-time threat detection and automated incident response. This system enables:

- Continuous monitoring of endpoints and network activity.
- Automated enrichment and triage using Shuffle workflows.
- Seamless case management via integrations with platforms like TheHive.
- Reduced analyst fatigue through alert prioritization and auto-response.

This architecture supports both small-scale deployments and enterprise-grade SOC environments, offering flexibility, transparency and extensibility.

ATPENDIX


Workflow - Wazuh
New Tab
192.168.100.55:3001/workflows/0181884f-816b-48cc-ab61-d762fd94e53f?execution_id=221dff2c-cf66-413c-b68
Workflows
Apps
Docs
Search Apps, Workflows, [
Search Active Apps
Shuffle Tools
Http
Email
VulnDB
Yara
Sooty
Thehive
Testing
Apps
Triggers
Vars
Workflows › Wazuh
Webhook
Execution Argu
All Workflow Runs
REFRESH RUNS
02/08/2024, 04:23:29 1 / 0 / 1
02/08/2024, 04:23:28 1 / 0 / 1
02/08/2024, 04:23:06 1 / 0 / 1
02/08/2024, 04:20:50 1 / 0 / 1
02/08/2024, 04:20:47 1 / 0 / 1
02/08/2024, 04:18:44 1 / 0 / 1
02/08/2024, 04:18:39 1 / 0 / 1
02/08/2024, 04:18:34 1 / 0 / 1
Welcome to Shuffle! We're one of the world's leading providers of no-code automation. How can we...


All Workflow Runs
REFRESH RUNS
02/08/2024, 04:23:29 1 / 0 / 1
02/08/2024, 04:23:28 1 / 0 / 1
02/08/2024, 04:23:06 1 / 0 / 1
02/08/2024, 04:20:50 1 / 0 / 1
02/08/2024, 04:20:47 1 / 0 / 1
02/08/2024, 04:18:44 1 / 0 / 1
02/08/2024, 04:18:39 1 / 0 / 1
02/08/2024, 04:18:34 1 / 0 / 1
Welcome to Shuffle! We're one of the world's leading providers of no-code automation. How can we...

a

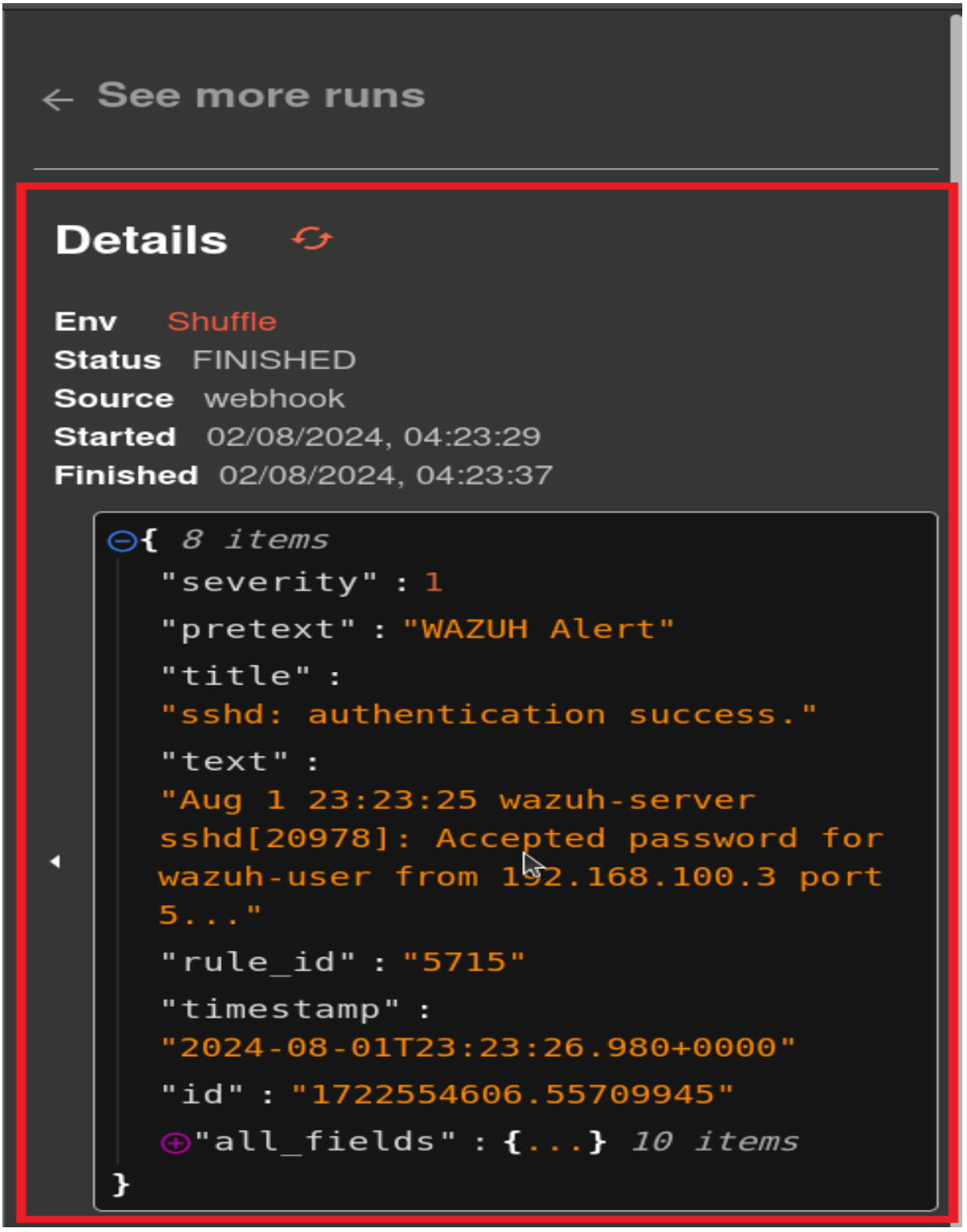

See more runs
Details
Env Shuffle
Status FINISHED
Source webhook
Started 02/08/2024, 04:23:29
Finished 02/08/2024, 04:23:37
{ 8 items
"severity" : 1
"pretext" : "WAZUH Alert"
"title" :
"sshd: authentication success."
"text" :
"Aug 1 23:23:25 wazuh-server
sshd[20978]: Accepted password for
wazuh-user from 192.168.100.3 port
5..."
"rule_id" : "5715"
"timestamp" :
"2024-08-01T23:23:26.980+0000"
"id" : "1722554606.55709945"
"all_fields" : {...} 10 items
}

Table JSON

t _index	wazuh-alerts-4.x-2024.08.01
t agent.id	000
t agent.name	wazuh-server
t data.dstuser	wazuh-user
t data.uid	0
t decoder.name	pam
t decoder.parent	pam
t full_log	Aug 1 23:23:26 wazuh-server sshd[20978]: pam_unix(sshd:session): session opened for user wazuh-user by (uid=0)
t id	1722554606.55709508
t input.type	log
t location	/var/log/secure
t manager.name	wazuh-server
t predecoder.hostname	wazuh-server
t predecoder.program_name	sshd

t manager.name	wazuh-server
t predecoder.hostname	wazuh-server
t predecoder.program_name	sshd
t predecoder.timestamp	Aug 1 23:23:26
t rule.description	PAM: Login session opened.
# rule.firedtimes	1
t rule.gdpr	IV_32.2
t rule.gpg13	7.8, 7.9
t rule.groups	pam, syslog, authentication_success
t rule.hipaa	164.312.b
t rule.id	5501
# rule.level	3
rule.mail	false
t rule.mitre.id	T1078
t rule.mitre.tactic	Defense Evasion, Persistence, Privilege Escalation, Initial Access
t rule.mitre.technique	Valid Accounts

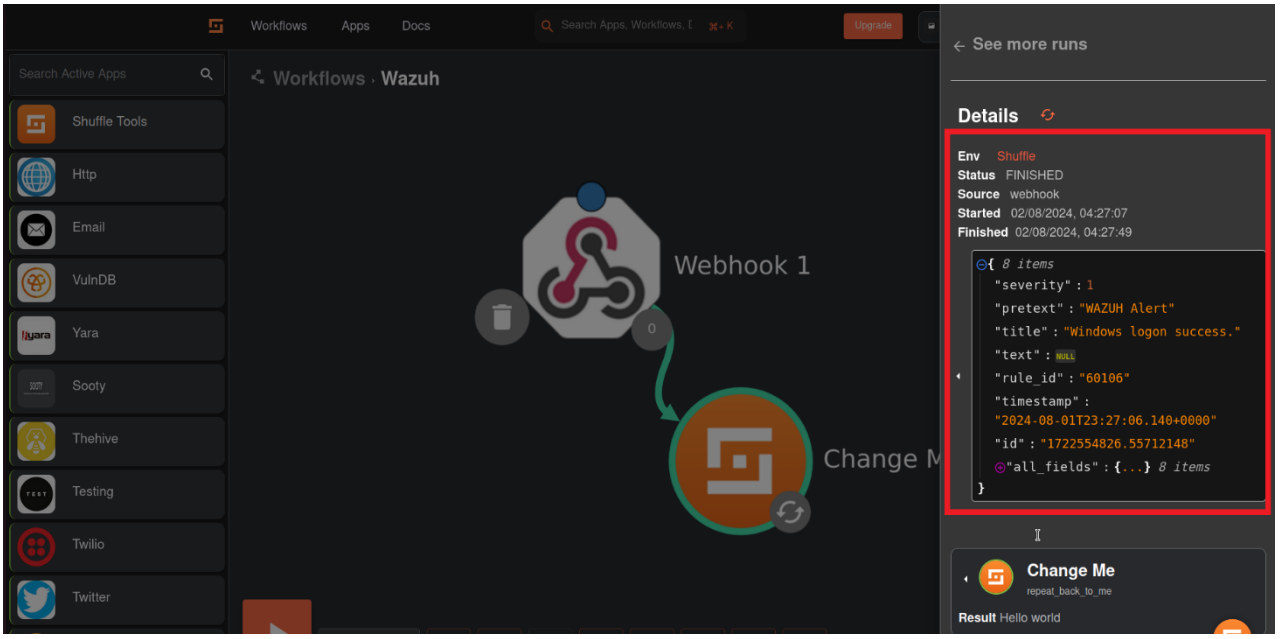
Aug 2, 2024 @ 04:27:06.604	Windows10-Sami	Software protection service scheduled successfully.	3	60642
Aug 2, 2024 @ 04:27:06.465	Windows10-Sami	Windows logon success.	3	60106
Aug 2, 2024 @ 04:27:06.449	Windows10-Sami	Windows logon success.	3	60106
Aug 2, 2024 @ 04:27:06.435	Windows10-Sami	Windows logon success.	3	60106
Aug 2, 2024 @ 04:27:06.405	Windows10-Sami	Windows logon success.	3	60106
Aug 2, 2024 @ 04:27:06.387	Windows10-Sami	Windows logon success.	3	60106
Aug 2, 2024 @ 04:27:06.372	Windows10-Sami	Windows logon success.	3	60106

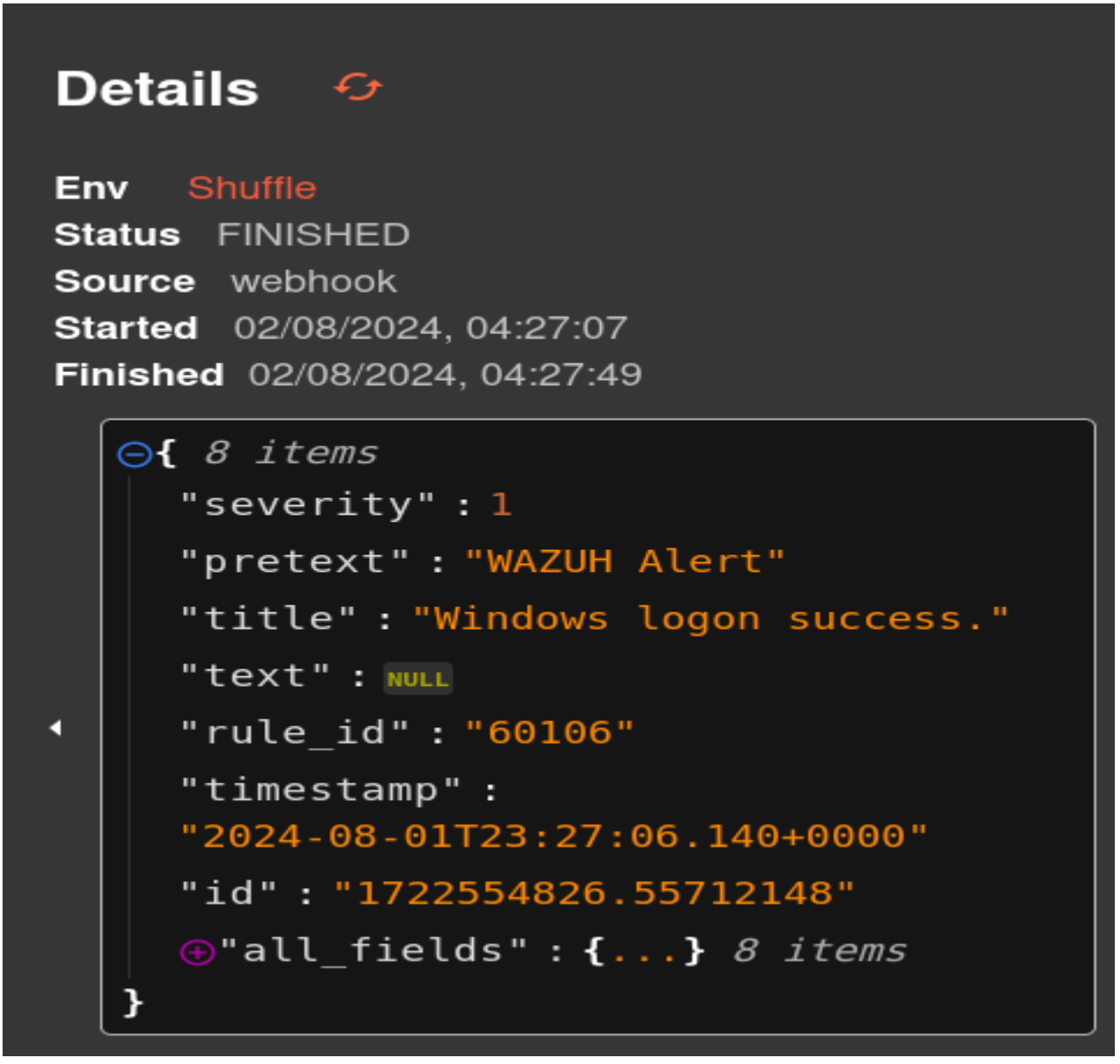
Aug 2, 2024 @ 04:27:06.465 Windows10-Sami Windows logon success. 3

Expanded document View surrounding documents

Table JSON

t _index	wazuh-alerts-4.x-2024.08.01
t agent.id	008
t agent.ip	192.168.100.11
t agent.name	Windows10-Sami
t data.win.eventdata.authenticationPackageName	Negotiate
t data.win.eventdata.elevatedToken	%%1842
t data.win.eventdata.impersonationLevel	%%1833
t data.win.eventdata.keyLength	0
t data.win.eventdata.logonGuid	{00000000-0000-0000-0000-000000000000}
t data.win.eventdata.logonProcessName	Advapi
t data.win.eventdata.logonType	5
t data.win.eventdata.processId	0x398


Workflows
Apps
Docs
Search Apps, Workflows, [
⌘+K
Upgrade
Search Active Apps
Shuffle Tools
Http
Email
VulnDB
Yara
Sooty
Thehive
Testing
Twilio
Twitter
Workflows › Wazuh
Webhook 1
0
Change M
See more runs
Details
Env Shuffle
Status FINISHED
Source webhook
Started 02/08/2024, 04:27:07
Finished 02/08/2024, 04:27:49
{ 8 items
"severity" : 1
"pretext" : "WAZUH Alert"
"title" : "Windows logon success."
"text" : NULL
"rule_id" : "60106"
"timestamp" :
"2024-08-01T23:27:06.140+0000"
"id" : "1722554826.55712148"
"all_fields" : {...} 8 items
}
Change Me
repeat_back_to_me
Result Hello world


Details
Env Shuffle
Status FINISHED
Source webhook
Started 02/08/2024, 04:27:07
Finished 02/08/2024, 04:27:49
{ 8 items
"severity" : 1
"pretext" : "WAZUH Alert"
"title" : "Windows logon success."
"text" : NULL
"rule_id" : "60106"
"timestamp" :
"2024-08-01T23:27:06.140+0000"
"id" : "1722554826.55712148"
"all_fields" : {...} 8 items
}

REFERENCES

1.Sharma and V. Gupta, “A comparative study of open-source SIEM tools for real- time threat detection,” International Journal of Computer Applications, vol. 182, no. 12, pp. 25–30, 2023.

2.Singh, R. Verma and T. Kaur, “Automated incident response using Shuffle and open-source threat intelligence,” International Journal of Cybersecurity Intelligence & Cybercrime, vol. 5, no. 1, pp. 45–52, 2024.

3.Wazuh Documentation. “Wazuh: The Open Source Security Platform.” [Online]. Available: https://documentation.wazuh.com/

4. Shuffle SOAR. “Low-Code Security Automation for Everyone.” [Online]. Available: https://shuffler.io/

5.NIST Special Publication 800-207, “Zero Trust Architecture,” National Institute of Standards and Technology, Aug. 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-207

6.A. Kumar and S. Singh, “Implementation of SIEM Using Open Source Tools for Real-Time Threat Detection,” International Journal of Advanced Computer Science and Applications, vol. 14, no. 2, pp. 45–52, 2024.

7.M. O’Connor, “Cybersecurity Threat Detection and Response Using Wazuh and Open Source Tools,” National College of Ireland Thesis Repository, 2023.

8.S. Raj and P. Mehta, “Designing a Security Operations Center Using Wazuh, Shuffle and TheHive,” JETIR, vol. 11, no. 5, pp. 120–126, May 2024.

9.IBM Security, “The Evolution of Zero Trust and the Frameworks That Guide It,” IBM Think Insights, 2024. [Online]. Available: https://www.ibm.com/think/insights/the-evolution-of-zero-trust-and-the-frameworks-that-guide-it

10.A. Sharma, “Exploring the Implementation and Challenges of Zero Trust Security Models,” IJERT, vol. 14, no. 5, pp. 146–152, 2025. [Online]. Available: https://www.ijert.org/research/exploring-the-implementation-and-challenges-of-zero-trust-security-models-in-modern-network-environments-IJERTV14IS050146.pdf.

11.Wazuh blog: “Wazuh and Shuffle Announce Technology Partnership” — describes the official partnership and integration of Wazuh XDR/SIEM with Shuffle SOAR.
Wazuh

12.Wazuh blog: “Integrating Wazuh with Shuffle” — detailed guide on how to configure Wazuh (ossec.conf) to send alerts to a Shuffle webhook, and build workflows.
Wazuh

13.Wazuh documentation: External API integration — describes how to configure the Integrator module to connect Wazuh with Shuffle via webhook.
documentation.wazuh.com

14.Wazuh documentation: ossec.conf integration options — the reference manual for setting <integration> blocks for services including Shuffle.
documentation.wazuh.com

15.GitHub — Divyansh121699 / SOC-Automation-Lab — an open-source lab project combining Wazuh, Shuffle, and TheHive to detect e.g. mimikatz activity.
GitHub

16.GitHub — uruc / SOC-Automation-Lab — similar SOC automation using Wazuh + Shuffle + TheHive, with detailed setup.
GitHub

17.GitHub — omarmalas / Automated-Security-Operations-with-Shuffle-SOAR — integrates Wazuh, Shuffle, TheHive to automate threat detection and response.
GitHub

18.GitHub — malwarekid / SOAR-Flow — Shake up incident response: Wazuh alerts + enrichment via VirusTotal & AbuseIPDB + case creation in TheHive + Discord notifications.
GitHub

19.International Journal Article — “Automated Defense Against Application-Layer Attacks on Windows Systems Using Wazuh and Shuffle” by Aastha Thakker, Aditya More, Kapil Kumar — academic framework combining Wazuh and Shuffle for app-layer threat detection.
lamintang.org