

Project Citadel: Next-Generation Security Analytics & Response Platform

Subsystem: Security Information and Event Management (SIEM)

Document Type: Subsystem Implementation Document

1. Executive Summary

This document outlines the implementation of the **Security Information and Event Management (SIEM)** layer within the **Citadel MVP** environment. The SIEM subsystem serves as the **central telemetry, detection, and analysis backbone**, enabling SOC to ingest diverse security logs, detect anomalies and attacks in real time, and visualize multi-tenant activity across infrastructure, applications, and Kubernetes workloads.

The Citadel SIEM architecture is built on a **hybrid Wazuh-ELK stack**, combining endpoint and network telemetry from multiple tenants with centralized indexing, visualization, and detection capabilities. The design emphasizes **multi-layer coverage, tenant separation, and scalability**.

Key outcomes delivered under this MVP include:

- **Multi-Layer Log Ingestion** - Telemetry from endpoints (Wazuh), network IDS (Suricata), application authentication (Keycloak), and Kubernetes workloads is collected and centralized in Elasticsearch for both Tenant 1 and Tenant 2.
- **Detection Rule Engineering** - Custom detection rules have been implemented for SSH failed logins, Keycloak authentication failures (per application and machine), and Kubernetes pod monitoring, ensuring targeted visibility across layers.
- **Multi-Tenant Coverage** - Separate ingestion, rules, and dashboards have been established for Tenant 1 and Tenant 2, allowing isolated analysis and faster triage of tenant-specific incidents.
- **Dashboards & Visualization** - Multiple Kibana dashboards provide analysts and stakeholders with real-time operational visibility, including SSH activity, authentication patterns, Suricata alerts, and Kubernetes events.
- **Scalable & Extensible Architecture** - The decoupled Wazuh and ELK deployments, with Elastic Fleet integration, enable future horizontal scaling and smooth onboarding of additional telemetry sources.

The SIEM subsystem **directly supports SOC objectives** by delivering unified visibility across critical telemetry streams, enabling precise detection logic, and establishing a centralized analysis layer that will integrate with SOAR automation.

2. Introduction

The **Security Information and Event Management (SIEM)** subsystem forms the **core detection and visibility layer** of the Citadel MVP environment. Its primary purpose is to **centralize security telemetry**, enable **real-time detection** of malicious activity, and provide **multi-tenant situational awareness** for SOC analysts and stakeholders.

Traditional security monitoring often suffers from fragmented telemetry, siloed detection rules, and limited scalability across diverse environments. Citadel's SIEM architecture addresses these challenges by integrating **endpoint, network, authentication, and Kubernetes telemetry** into a unified Elastic-based analytics platform with **tenant-aware ingestion, detection, and visualization**.

The Citadel MVP charter outlined clear SIEM objectives to be achieved in the initial phase:

- **Centralized Log Ingestion**

Ingest security logs from multiple layers - endpoints, network IDS, authentication systems, and Kubernetes environments - into a central SIEM platform for both Tenant 1 and Tenant 2.

- **Multi-Tenant Visibility**

Maintain clear separation of telemetry, rules, and dashboards for multiple tenants to support accurate incident triage and reporting.

- **Detection Coverage**

Implement core detection rules for SSH brute force attempts, authentication failures, and container activity to identify key attack vectors early.

- **Real-Time Dashboards**

Develop Kibana dashboards for analysts and stakeholders, enabling immediate insight into SSH activity, authentication events, IDS alerts, and container behavior.

Implementation Overview

To meet these requirements, a **hybrid SIEM architecture** was deployed, consisting of:

- **Wazuh Server** (Dedicated Machine)
 - Endpoint monitoring and security event collection.
 - Hosts the Elastic Agent Fleet Server for endpoint onboarding and telemetry forwarding.
- **ELK Stack (Elasticsearch, Logstash, Kibana)** (Separate Machine)
 - Central log indexing, visualization, and detection pipeline.
 - Receives telemetry from Elastic Agents, Wazuh, Suricata IDS, Keycloak authentication logs, and Kubernetes workloads.
- **Tenant Endpoints** (Tenant 1 and Tenant 2)
 - Both tenants have Wazuh agents and Elastic Agents installed.
 - Suricata IDS is deployed and integrated to forward network alerts.
 - Two Keycloak-protected web applications per tenant send authentication logs to ELK.
- **Kubernetes Cluster**
 - Elastic Agent is deployed in the cluster to collect container and pod logs.
 - Kubernetes telemetry enables pod monitoring and security event detection.

Detection Engineering

Custom detection rules have been developed for:

- **SSH Failed Logins** - Separate rules for Tenant 1 and Tenant 2 to detect brute force or credential misuse attempts.
- **Keycloak Authentication Failures** - Separate per application and per machine, enabling fine-grained monitoring of authentication anomalies.
- **Kubernetes Pod Monitoring** - Detects pod lifecycle anomalies or unexpected behavior using container logs.

These rules form the **first layer of behavioral and signature-based detection**, ensuring coverage across critical telemetry streams.

Visualization & Analysis

Multiple **Kibana dashboards** were created to visualize tenant-specific telemetry, including:

- SSH activity timelines and failure patterns

- Authentication events from Keycloak
- Suricata IDS alerts (e.g., port scans, exploitation attempts)
- Kubernetes pod activity and workload metrics

Dashboards are **segregated by tenant**, ensuring clarity during investigations and avoiding data leakage between environments.

Strategic Role within Citadel MVP

The SIEM layer acts as the **central nervous system** of Citadel MVP:

- **Feeds UEBA** with authentication and telemetry logs for behavioral anomaly detection.
- **Triggers SOAR workflows** in future phases based on detection rules and alert outputs.
- **Provides visibility** into both traditional (VM-based) and cloud-native (Kubernetes) environments.

This unified telemetry and detection layer lays the foundation for a **scalable, multi-tenant SOC platform** that aligns with long-term Zero Trust and SOC modernization strategies.

3. Scope

This section defines the **current functional scope, operational constraints, identified gaps, and future enhancement areas** for the SIEM subsystem deployed under Citadel MVP. It establishes a clear view of the telemetry coverage, detection capabilities, and analysis workflows currently implemented, along with planned extensions for future phases.

3.1 Current Scope

The SIEM subsystem currently provides **multi-layer telemetry ingestion, tenant-aware detection rules, and operational dashboards** across the following components:

Telemetry Sources

- **Endpoint Security (Wazuh)**
 - Log collection from Tenant 1 and Tenant 2 VMs through Wazuh agents.
 - Security events, file integrity monitoring, and syslog forwarding.
- **Network Monitoring (Suricata IDS)**
 - IDS alerts from both tenants' Suricata deployments.

- Coverage includes reconnaissance, brute force, and exploitation signatures.
- **Authentication Logs (Keycloak)**
 - Centralized ingestion of authentication events from Keycloak-managed web applications on each tenant machine.
 - Enables monitoring of failed logins and suspicious access behavior per application and machine.
- **Kubernetes Workloads (Elastic Agent)**
 - Container logs and pod lifecycle events are collected via an Elastic Agent deployed in the cluster.
 - Enables pod monitoring and workload-level visibility.

Detection Rules

- **SSH Failed Login Attempts**
 - Separate rules per tenant to detect brute force or repeated failed access attempts.
- **Keycloak Authentication Failures**
 - Separate rules per application and per machine to detect repeated or suspicious login failures.
- **Pod Monitoring Rule**
 - Detects unexpected or abnormal pod behavior in the Kubernetes environment.

Dashboards & Analysis

- Multiple **Kibana dashboards** have been created to visualize:
 - SSH login activity trends
 - Keycloak authentication failure timelines
 - Suricata IDS alerts per tenant
 - Kubernetes pod activity
- Dashboards are **segregated by tenant**, enabling analysts to investigate incidents independently without telemetry overlap.

Tenant Coverage

- **Tenant 1 and Tenant 2** are fully onboarded, with separate telemetry pipelines, detection rules, and dashboards.

- This ensures clear multi-tenant visibility and operational boundaries within the same SIEM deployment.

3.2 Constraints

The following constraints apply to the current SIEM MVP:

- **Single-Node ELK Deployment**
 - Elasticsearch, Logstash, and Kibana are hosted on a single machine, without HA or clustering.
- **Basic Parsing & Normalization**
 - Log parsing and ECS mapping are functional but not yet fully normalized for all custom fields.
- **Limited Retention & Scaling**
 - Log retention is limited to MVP timelines; scaling for long-term storage or high ingest volumes is not yet configured.
- **Manual Rule Tuning**
 - Detection rules use static thresholds and are manually maintained; no adaptive tuning or rule versioning is implemented.

3.3 Gaps Identified

- **Coverage Gaps**
 - No cloud-native telemetry sources beyond Kubernetes (e.g., Azure/AWS logs) are ingested at this stage.
 - No UEBA enrichment is applied to SIEM detections yet. A separate UEBA model is used with inbuild dashboards and direct automation with threat intelligence.
- **Detection Depth**
 - Detection rules are currently focused on failed logins and basic Kubernetes monitoring; advanced correlation rules are pending.
- **Limited Historical Analysis**
 - Due to retention limits, long-range threat hunting or trend analysis is not yet supported.
- **No Dedicated Multi-Tenant RBAC**

- Dashboards and rules are segregated logically, but fine-grained role-based access controls are not yet configured in Kibana.

3.4 Future Enhancements

The following enhancements are planned for future Citadel phases:

- **Scalable & Redundant ELK Architecture**
 - Implement Elasticsearch clustering and multi-node Kibana for high availability and larger ingestion volumes.
- **Expanded Telemetry Sources**
 - Ingest Azure, AWS, and application-layer logs to broaden detection coverage.
 - Integrate Falco runtime alerts for Kubernetes security.
- **Detection Rule Expansion**
 - Add correlation rules for lateral movement, brute-force chaining, privilege escalation, and cross-layer attack patterns.
- **Advanced Dashboards & RBAC**
 - Implement role-based views, multi-tenant RBAC, and enriched dashboards for SOC teams and stakeholders.

4. Tools, Components & Solution Overview

This section provides a structured overview of the **tools**, **core components**, and **solution architecture** that form the foundation of the SIEM subsystem in Citadel MVP. The architecture was designed to achieve **multi-layer telemetry ingestion**, **multi-tenant visibility**, and **scalable detection capabilities** while maintaining operational simplicity suitable for an MVP phase.

4.1 Core SIEM Components

Component	Role
Wazuh Server	Collects endpoint logs, security events, and FIM data from tenant machines; hosts the Elastic Fleet Server for agent management.
Elastic Stack	Elasticsearch and Kibana deployed on a dedicated machine for centralized log storage, indexing, visualization, and detection.

Elastic Agent	Deployed on tenant VMs and Kubernetes nodes to forward system logs, application telemetry, and container events to ELK.
Suricata IDS	Network intrusion detection system deployed per tenant, generating alerts ingested into ELK.
Keycloak	Provides authentication logs from tenant web applications, enabling detection of failed login patterns and suspicious activity.
Kubernetes Agent	Elastic Agent deployed inside Kubernetes cluster to collect container logs, pod events, and workload telemetry.
Kibana Dashboards	Visualization layer for SOC analysts and stakeholders, supporting tenant-specific dashboards for SSH, Keycloak, Suricata, and Kubernetes telemetry.

This combination of **endpoint, network, authentication, and container telemetry** provides the SOC with comprehensive visibility across both traditional VM-based and cloud-native environments.

4.2 Log Ingestion Architecture

The Citadel SIEM follows a **multi-source ingestion pipeline** designed for tenant isolation and centralized visibility:

1. Tenant Endpoint Logs

- Wazuh agents and Elastic Agents on Tenant 1 and Tenant 2 VMs collect syslog, security, and endpoint telemetry.
- Logs are sent to Wazuh and then forwarded to ELK via Fleet.

2. Network Telemetry

- Suricata IDS instances run on each tenant VM, generating alerts for reconnaissance, brute force, and exploitation activity.
- These alerts are ingested into ELK for centralized indexing and detection.

3. Application Authentication Logs

- Keycloak manages authentication for two web applications per tenant.
- Authentication logs (e.g., login successes, failures, error states) are forwarded to ELK for analysis and failed-login detections.

4. Kubernetes Logs & Events

- An Elastic Agent is deployed inside the Kubernetes cluster to monitor workloads, collect container logs, and observe pod lifecycle events.
- This allows visibility into containerized service behavior and cluster-level anomalies.

4.3 Detection & Analysis Components

Component	Function
Elasticsearch	Custom queries and detection rules to monitor SSH failed logins, Keycloak authentication failures, and Kubernetes pod activity.
Tenant-Specific Segregation	Rules are scoped separately for Tenant 1 and Tenant 2 to ensure isolation and accurate incident triage.
Dashboards	Multiple Kibana dashboards visualize security events and telemetry per tenant, supporting analysis and incident investigation.
Suricata Signatures	IDS signatures detect common attack behaviors such as brute force, port scans, and exploit attempts.

Detection logic leverages **both endpoint logs and network signals** to identify key security events early in the attack chain.

4.4 Tenant Architecture

Citadel MVP is designed as a **multi-tenant SIEM**, supporting **Tenant 1** and **Tenant 2** through logically segregated pipelines:

- **Separate Agent Deployments:** Each tenant has independent Wazuh and Elastic Agents.
- **Separate Detection Rules:** SSH and authentication rules are scoped per tenant to avoid cross-tenant interference.
- **Tenant Dashboards:** Dashboards are filtered and indexed per tenant for isolated visibility.
- **Centralized Indexing:** Both tenants share the same Elasticsearch cluster but are logically separated through index patterns and filters.

This structure allows the SOC to manage multiple environments within a single SIEM platform **without sacrificing clarity or control**.

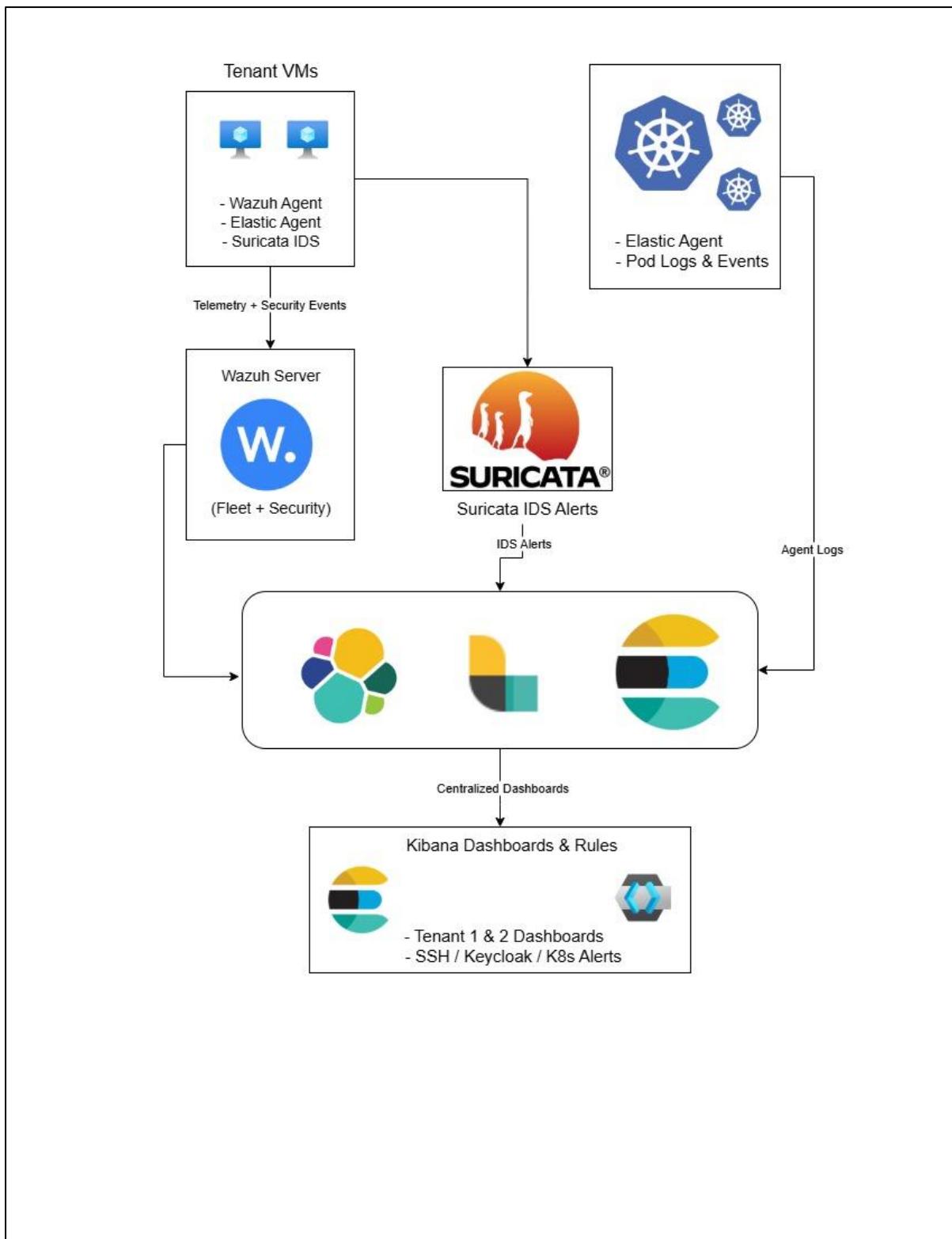
4.5 Kubernetes Monitoring Integration

The inclusion of Kubernetes telemetry is a **strategic differentiator** in the Citadel MVP SIEM design:

- **Elastic Agent** runs inside the cluster, forwarding container logs and pod lifecycle events.
- **Pod monitoring rules** detect anomalous events or unexpected workload behaviors.
- Kubernetes logs are visualized in **dedicated dashboards**, complementing VM-based telemetry.

This integration aligns with goal of **bridging traditional SOC visibility with cloud-native environments**, supporting future Zero Trust and CNAPP strategies.

4.6 Solution Architecture Diagram



5. Configurations & Customizations

This section outlines the **key configurations and environment-specific customizations** implemented during the deployment of the Citadel MVP SIEM subsystem. These

configurations ensure that telemetry is reliably ingested, correctly normalized, and accurately analyzed across multiple tenants, telemetry sources, and Kubernetes workloads.

5.1 Wazuh Server Configuration

The **Wazuh server** plays a dual role:

- As a **security event collector**, receiving logs from Tenant 1 and Tenant 2 endpoints.
- As the **Elastic Fleet Server**, managing Elastic Agent enrollment and policy distribution.

Key Configurations:

- **Agent Enrollment:**
 - Each tenant's endpoints were enrolled separately, generating unique keys to maintain tenant boundary control.
 - Wazuh agents were configured to forward endpoint logs, syslog, and security events.
- **Fleet Server:**
 - Fleet server was integrated with the ELK.
 - TLS and authentication were configured to secure agent communications.
- **Integration with Suricata:**
 - Wazuh was configured to receive Suricata alerts from local IDS instances, ensuring network telemetry is available for correlation.

5.2 Elastic Stack (ELK) Configuration

The ELK Stack provides the **central indexing and detection layer**. Deployed on a single dedicated machine for MVP, it was tuned to handle multiple telemetry streams efficiently.

Component	Key Configurations
Elasticsearch	- Configured with index templates to maintain consistency across tenant indices. - Mappings adjusted for geo/IP fields (from Suricata and Keycloak) to

	<p>support search and dashboards.</p> <ul style="list-style-type: none"> - Basic retention policies set to match MVP scope.
Kibana	<ul style="list-style-type: none"> - Multi-tenant index patterns defined for Tenant 1 and Tenant 2. - Dashboards and visualizations segregated by tenant. - Detection rules configured for SSH failures, Keycloak auth failures, and pod monitoring.
Elastic Agent	Agent Logs Forwarding

Notable Customizations:

- GeoIP enrichment for Suricata alerts to support geographic dashboards.
- Parsing adjustments for Keycloak authentication fields to ensure failed login events are properly classified.
- Custom index naming conventions to clearly differentiate between telemetry sources and tenants.

5.3 Elastic Agent Policies

Elastic Agents were deployed on:

- **Tenant 1 and Tenant 2 endpoints (VMs)**
- **Inside the Kubernetes cluster**

Policies & Customizations:

- **System Logs & Metrics:**

Agents were configured to forward syslog, process, and host metrics to Elasticsearch.

- **Suricata Integration:**

Suricata log directories were added as custom log paths in the Elastic Agent policy, ensuring IDS alerts are ingested alongside endpoint logs.

- **Kubernetes Policy:**

The Kubernetes agent was configured to collect:

- Container stdout/stderr logs
- Pod lifecycle events

- Node-level metrics

This enabled pod monitoring rules and cluster visibility dashboards.

- **Tenant Segregation:**

Policies were tagged by tenant to ensure telemetry separation at ingestion time.

5.4 Suricata IDS Configuration

Suricata was installed and configured separately on **Tenant 1** and **Tenant 2** VMs.

Key Customizations:

- **EVE JSON Output:**

Enabled EVE JSON logging to a dedicated file for Elastic Agent pickup.

- **Tuned Rule Sets:**

Default rule sets were augmented to include detection of SSH brute force, port scans, and basic exploitation attempts.

- **Forwarding:**

Logs were forwarded locally to Elastic Agent for ingestion into ELK. Suricata instances were configured to tag alerts with tenant identifiers at log generation time.

5.5 Keycloak Log Integration

Keycloak authentication logs from both tenants' web applications were ingested to support **failed login monitoring** and **application-level anomaly detection**.

Key Configurations:

- Application logs were shipped via Elastic Agent to ELK.
- Custom ECS mappings ensured compatibility with dashboard filters and detection rules.
- Tenant and application labels were appended during ingestion for fine-grained detection scoping.

5.6 Kubernetes Agent Configuration

An Elastic Agent was deployed inside the Kubernetes cluster to enable container and workload visibility.

Key Configurations:

- Agent DaemonSet used to cover all nodes in the cluster.
- Container logs collected from stdout/stderr, automatically labeled with namespace, pod name, and container name.
- Kubernetes API integration enabled collection of pod lifecycle events for anomaly detection.
- Dedicated index patterns created for Kubernetes logs to enable pod monitoring dashboards and detection rules.

5.7 Detection Rule Customizations

Custom detection rules were written in Kibana for **tenant-aware monitoring** of critical telemetry streams:

Rule Name	Source	Customization
SSH Failed Login – Tenant 1	Authentication logs	Filters based on tenant index and source IP; tuned threshold for failed attempts.
SSH Failed Login – Tenant 2	Authentication logs	Separate rule, same detection logic but scoped to Tenant 2 indices.
Keycloak Failed Login – T1 & T2	Keycloak logs	Separate rules per application and per machine; uses parsed event_type and error fields.
Kubernetes Pod Monitoring	K8s Agent logs	Detects unexpected pod lifecycle events; leverages Kubernetes namespace/container metadata.

All rules are **scoped by tenant and source**, ensuring clean operational separation and more precise detection coverage.

5.8 Dashboard Customizations

Multiple **tenant-specific dashboards** were created in Kibana to support real-time visualization and analysis:

- **SSH Activity Dashboards:**
Display timelines, failure trends, and top offending IPs per tenant.
- **Keycloak Authentication Dashboards:**
Show login failure patterns, affected applications, and user activity distribution.
- **Suricata Dashboards:**
Visualize network IDS alerts, including geographic distribution and top signatures per tenant.
- **Kubernetes Dashboards:**
Monitor pod events, container logs, and node metrics for cluster security visibility.

Dashboards use **filters and index patterns per tenant**, ensuring analysts can switch context cleanly between Tenant 1 and Tenant 2 environments.

6. Implementation Details & Workflows

This section describes the **end-to-end implementation** of telemetry ingestion, detection rule application, and analysis workflows in the Citadel MVP SIEM subsystem. The workflows combine **multi-source log ingestion**, **tenant-specific detection logic**, and **dashboard-driven analysis** to SOC analysts with comprehensive, real-time visibility.

6.1 Log Ingestion Workflow

The ingestion pipeline is structured to **aggregate multiple telemetry streams** from both tenants and Kubernetes into a centralized Elasticsearch backend.

Steps:

1. **Endpoint & System Logs (Tenant 1 & Tenant 2)**
 - Wazuh agents collect syslog, FIM events, and security telemetry from VM endpoints.
 - Tenant tags are appended during ingestion for logical separation.
2. **Suricata IDS Alerts**
 - Suricata generates EVE JSON logs for each tenant, capturing network intrusion events.
 - Elastic Agent monitors these files and forwards them to Logstash → Elasticsearch.

- GeoIP enrichment is applied, and alerts are tagged with tenant IDs for indexing.

3. Keycloak Authentication Logs

- Application logs from Keycloak-managed web apps (2 per tenant) are shipped via Elastic Agent.
- Logstash parsing filters normalize event structure (username, error type, client ID).
- Logs are indexed under tenant-specific patterns for detection rules and dashboards.

4. Kubernetes Telemetry

- An Elastic Agent DaemonSet collects container stdout/stderr logs, pod lifecycle events, and node metrics.
- These are sent directly to Elasticsearch and indexed under Kubernetes-specific index patterns.

Outcome:

All telemetry is **centralized in Elasticsearch with consistent schema, tenant tags, and index naming conventions**, enabling unified analysis and detection.

6.2 Detection Workflow

The detection layer is built using **Kibana detection rules** and **Suricata IDS signatures** to cover endpoint, network, authentication, and Kubernetes activity.

Key Detection Logic Implemented:

Detection Scenario	Source	Logic
SSH Failed Logins – Tenant 1	Authentication logs	Counts repeated failed SSH login attempts from the same IP/user within a short time window.
SSH Failed Logins – Tenant 2	Authentication logs	Same logic, scoped to Tenant 2's indices.
Keycloak Authentication Failures (per app/machine)	Keycloak logs	Filters events by event_type = LOGIN_ERROR and groups by application, user, and source IP.

Pod Monitoring Rule	Kubernetes logs/events	Detects unexpected pod restarts, deletions, or namespace anomalies using Elastic Agent Kubernetes data.
Suricata Network Alerts	IDS JSON logs	Signature-based alerts for brute force, scans, and exploitation attempts.

Tenant Isolation:

- Separate rules are defined for Tenant 1 and Tenant 2 wherever applicable.
- Index patterns, query filters, and tags ensure that detections remain logically separated.

6.3 Alert & Dashboard Integration Workflow

Detection results are **visualized in Kibana dashboards** and can be **escalated to alerts** (manual triage and SOAR will automate this).

Flow:

1. Detection Triggers

- Rules are evaluated on indexed data. When conditions are met, detection events are stored in .siem-signals indices.
- Suricata alerts flow directly into dashboards.

2. Dashboard Visualization

- Multiple dashboards provide **real-time situational awareness**:
 - **SSH Activity Dashboard:** Timeline of failed logins, top offending IPs, geolocation maps.
 - **Keycloak Dashboard:** Authentication failure trends per app and machine, user-level breakdowns.
 - **Suricata Dashboard:** Alert volume trends, top signatures, attacker geolocation.
 - **Kubernetes Dashboard:** Pod events, container logs, workload anomalies.

3. Analyst Triage

- SOC analysts use dashboards to pivot on fields such as source IP, username, pod name, or signature ID.

- Cross-tenant investigations are avoided by using dashboard filters and index isolation.

6.4 Tenant-Specific Workflows

Because Citadel supports **multi-tenant SOC operations**, **parallel ingestion–detection–analysis workflows** exist for Tenant 1 and Tenant 2:

- **Separate Wazuh & Elastic Agents** ensure logs are tagged correctly at source.
- **Dedicated detection rules** per tenant avoid noise and misattribution.
- **Tenant-specific dashboards** provide isolated visualizations for investigations.

Example:

- A surge of SSH failed logins on Tenant 2 does not affect Tenant 1 dashboards or detection alerts.
- Keycloak failed login rules trigger independently per application and tenant, enabling precise incident scoping.

6.5 Kubernetes Monitoring Workflow

The Kubernetes integration enables visibility into **containerized workloads**, which is often overlooked in MVP SOC setups.

Workflow:

1. **Elastic Agent DaemonSet** collects container logs and pod events.
2. **Pod Monitoring Rule** detects anomalies such as frequent pod restarts or namespace misuse.
3. **Kubernetes Dashboard** displays pod timelines, container logs, and node metrics, helping analysts detect workload issues or suspicious behavior.

This integration ensures **cloud-native visibility** aligns with VM-based telemetry, supporting Zero Trust and CNAPP objectives.

6.6 Analyst Investigation Workflow

SOC analysts follow a structured workflow for detection triage and analysis:

1. **Detection Event Appears**
 - o Triggered by rules (SSH, Keycloak, K8s) or Suricata alerts.
2. **Dashboard Investigation**
 - o Analysts open the relevant dashboard (tenant-specific) to examine surrounding context, timelines, and related events.
3. **IOC & Pattern Extraction**
 - o Analysts extract attacker IPs, usernames, geolocations, pod/container names, or Suricata signature IDs.
4. **Cross-Source Correlation**
 - o Correlate SSH failures with Suricata alerts or Keycloak login anomalies to identify coordinated attacks.

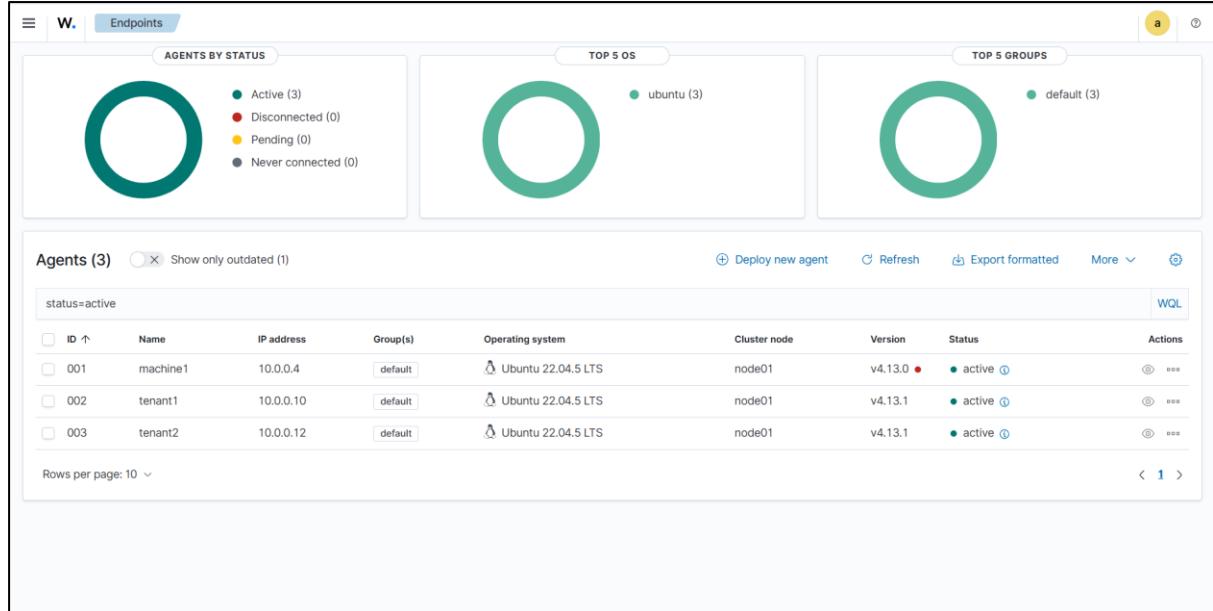
7. Evidence (Screenshots, Logs, Dashboards, Scripts)

This section presents **supporting evidence** validating the correct implementation and operation of the SIEM subsystem across telemetry ingestion, detection, and analysis layers. The evidence includes **screenshots, log extracts, dashboards, and detection outputs**, collected during the Citadel MVP deployment and validation phase.

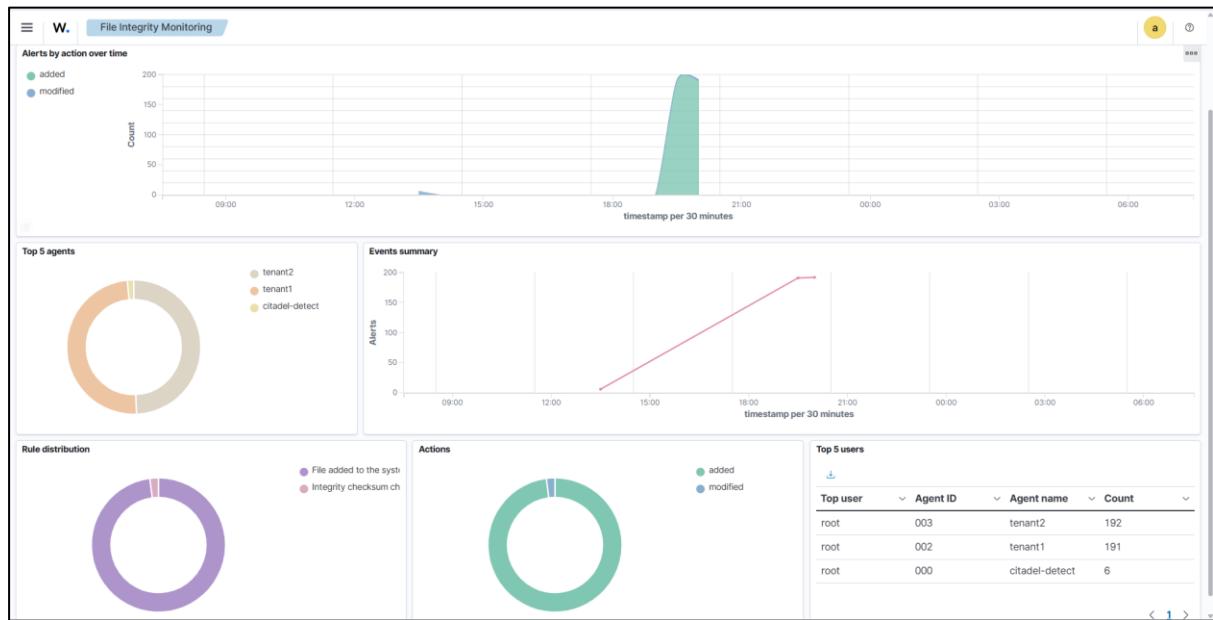
7.1 Telemetry Ingestion Validation

7.1.1 Endpoint & Wazuh Agent Ingestion

- Screenshots show successful agent registration for both Tenant 1 and Tenant 2 on the Wazuh server.



- Log extracts confirm syslog, file integrity monitoring (FIM), and security events.

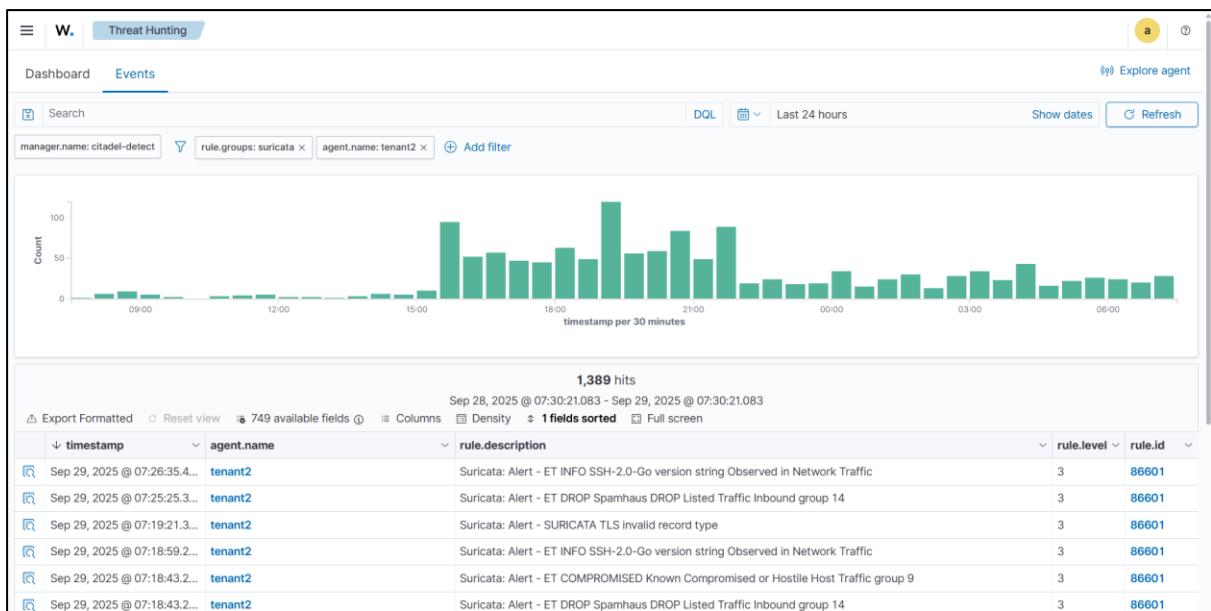
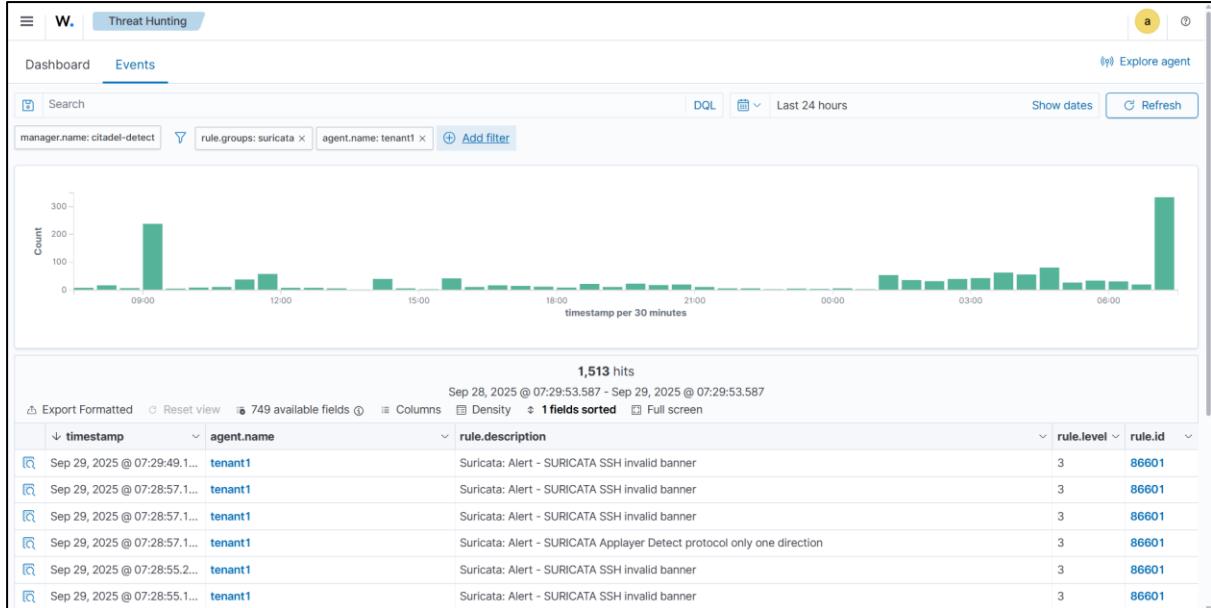


7.1.2 Suricata IDS Log Ingestion

- EVE JSON logs from Suricata are successfully picked up by Elastic Agent and forwarded to Elasticsearch.

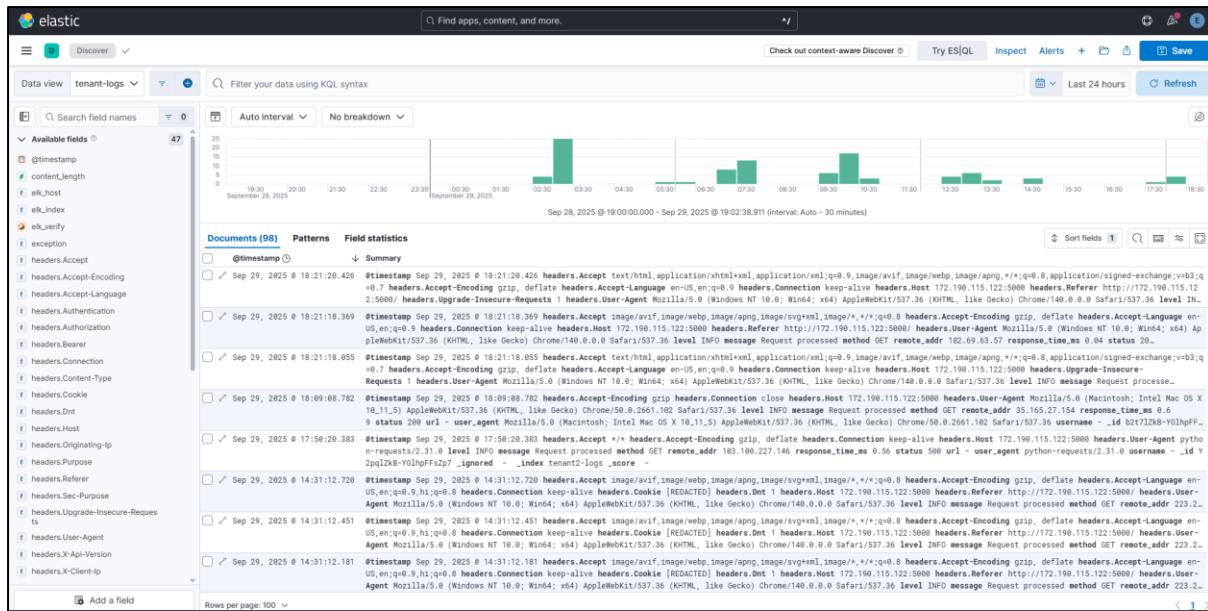
- Index inspection confirms fields such as `src_ip`, `dest_ip`, `alert.signature`, and `geoip.country_name` are properly parsed.

- EVE JSON logs from Suricata are successfully picked up by Wazuh agent and forwarded to Wazuh manager.



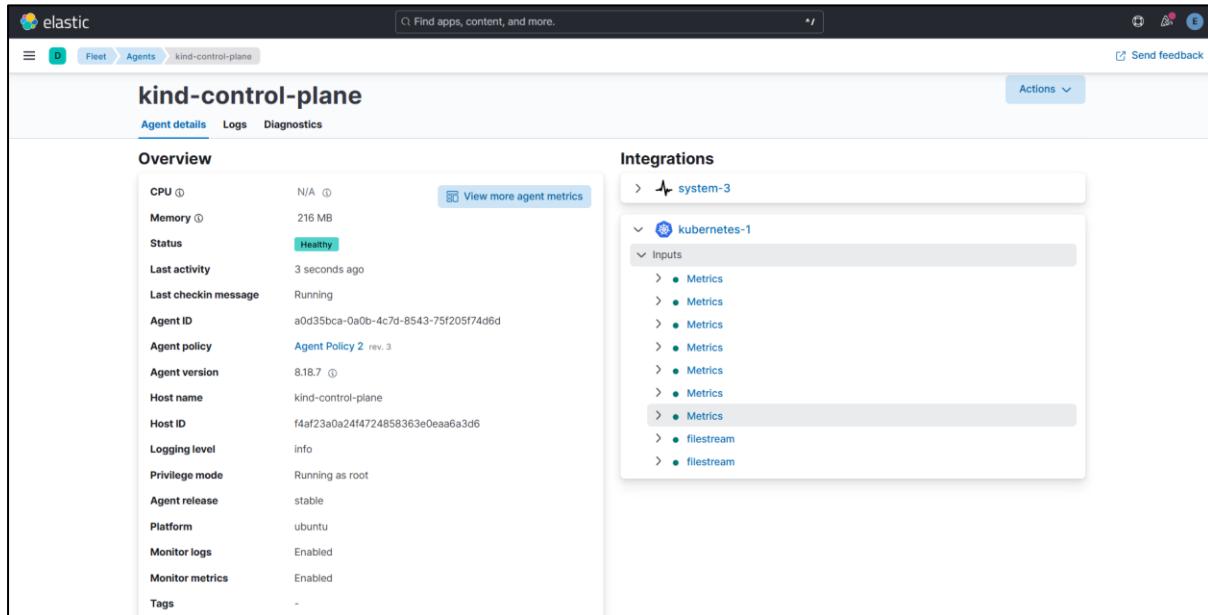
7.1.3 Keycloak Authentication Log Ingestion

- Authentication failure events from both tenants' web applications are correctly parsed and ingested.
- Grok and JSON filters normalize Keycloak logs for username, application ID, and error type.

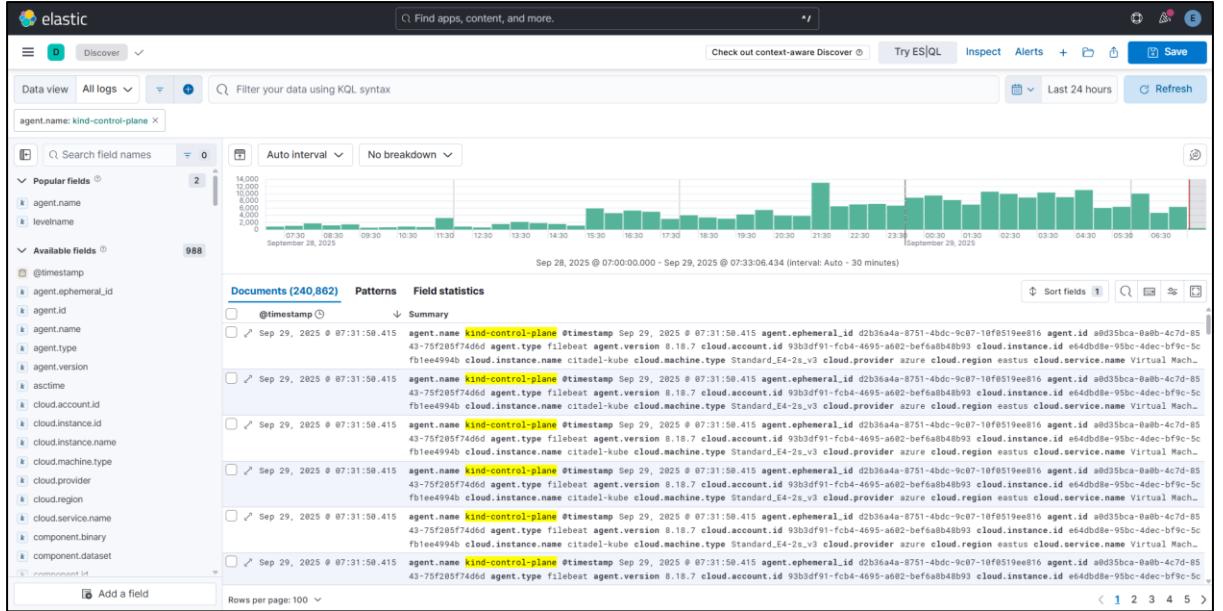


7.1.4 Kubernetes Telemetry Ingestion

- Elastic Agent DaemonSet deployed inside the cluster successfully collects container logs and pod lifecycle events.



- Kubernetes events (e.g., pod restarts) are visible in Elasticsearch and dashboards.



7.2 Detection Rule Validation

Detection rules were tested with simulated scenarios to ensure they trigger correctly for both tenants and Kubernetes workloads.

Rule	Scenario	Validation Evidence
SSH Failed Login – Tenant 1	Multiple failed SSH logins from a test IP	Kibana Detection UI shows triggered rule in .siem-signals
SSH Failed Login – Tenant 2	Repeated failed SSH attempts on Tenant 2 VM	Alert generated with Tenant 2 tag; visible in dashboard
Keycloak Failed Login – App1 (Tenant 1)	Injected failed login events through Keycloak logs	Rule fired correctly, dashboard updated
Keycloak Failed Login – App2 (Tenant 2)	Similar simulation for second tenant application	Independent rule triggered, isolated by tenant
Pod Monitoring Rule	Forced a pod crash/restart in the cluster	Rule detected unexpected pod event, visible in K8s dashboard
Suricata Network Alerts	Nmap scan and brute-force simulation	Suricata alerts ingested, signature matched, dashboard visualization confirmed

The screenshot shows the 'Rules' section of the Elastic Security interface. It lists six installed rules, all of which are currently enabled. The columns include Rule name, Risk score, Severity, Last run, Last response, Last updated, Notify, and Enabled status. The rules are:

- Brute Force Alert - Tenant 1
- Keycloak Failed Login - App1 (Tenant 1)
- Keycloak Failed Login - App2 (Tenant 2)
- Pod Monitoring Rule
- Suricata Network Alert
- Brute Force Alert - Tenant 2

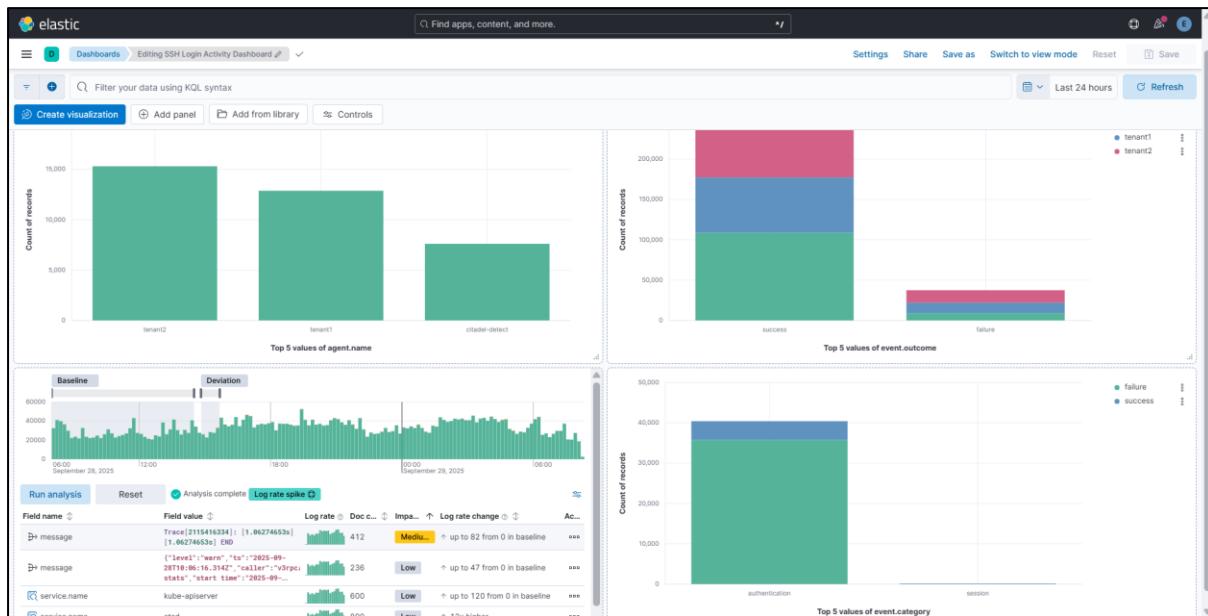
Each rule has a green success icon next to it, indicating they have succeeded. The last update for most rules was within the last few hours.

7.3 Dashboard Visualization Evidence

Multiple **Kibana dashboards** were validated to confirm real-time visualization of ingested telemetry and detection results:

- **SSH Login Activity Dashboard**

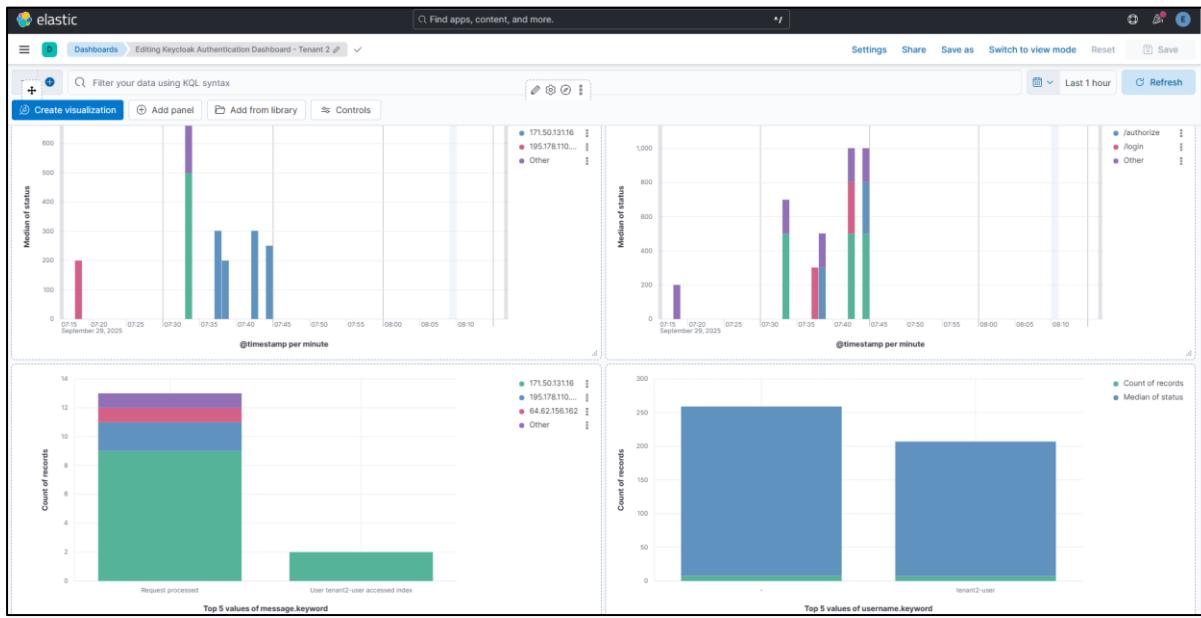
- Displays failed login timelines, top source IPs, geolocation maps per tenant.
- Tested with simulated brute force activity on both tenants.



- **Keycloak Authentication Dashboard**

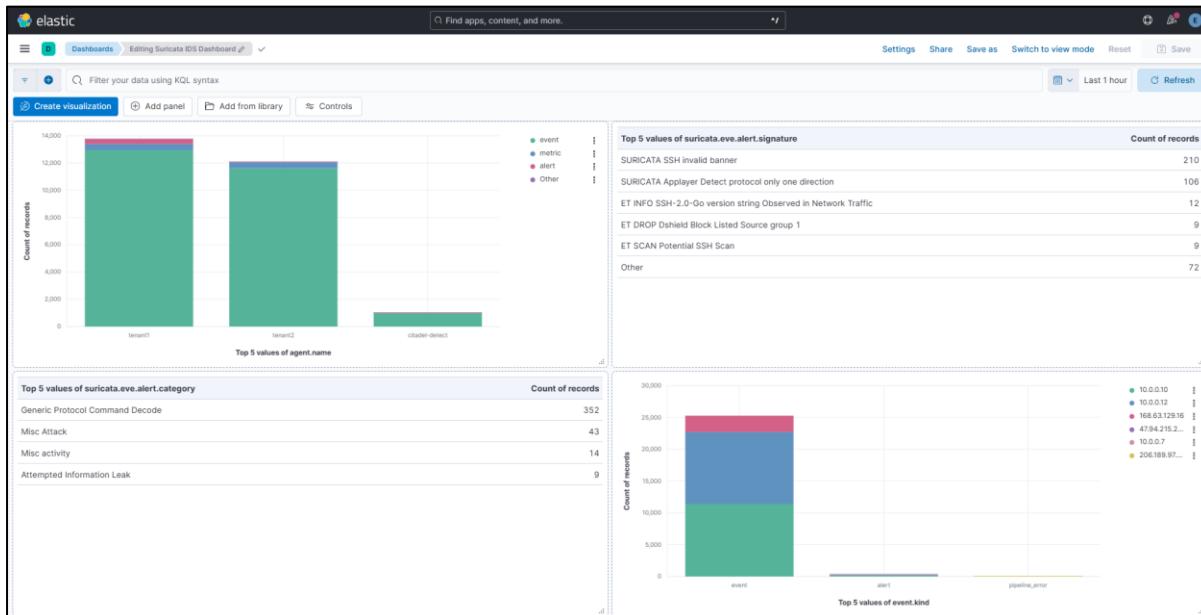
- Displays failed login trends segmented by application, user, and source IP.

- Verified tenant isolation via filters and index patterns.



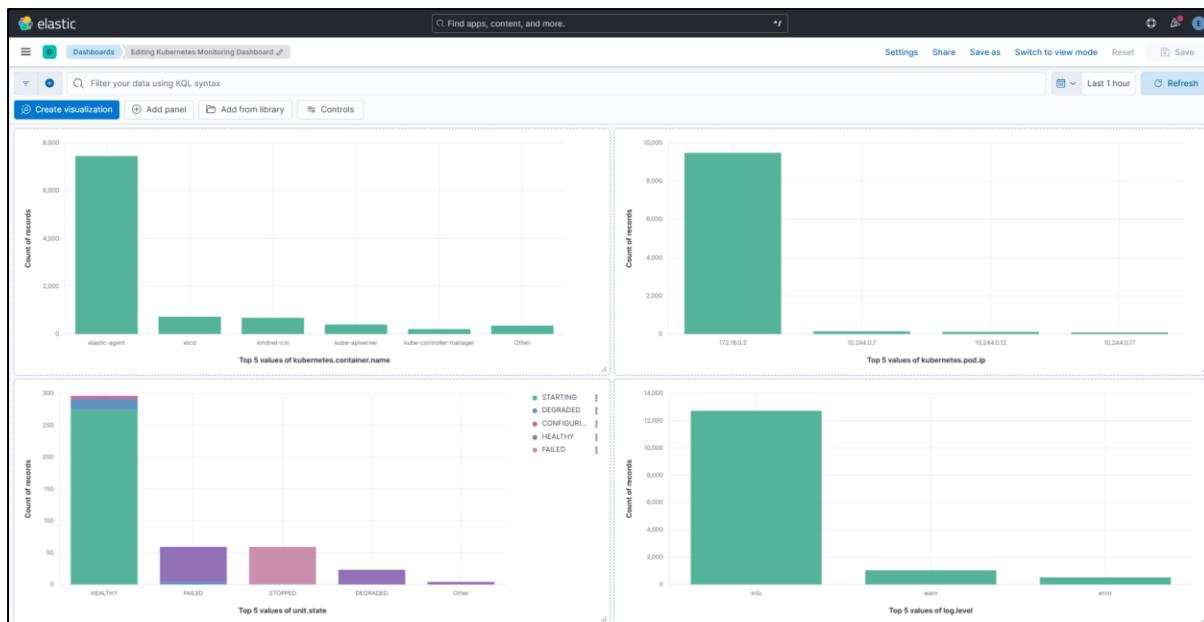
• Suricata IDS Dashboard

- Displays alert trends, top signatures, attacker countries, and IDS severity levels.
- Verified alerts for brute force and scan signatures.



• Kubernetes Monitoring Dashboard

- Displays pod event timelines, container logs, and node metrics.
- Verified with pod restart simulation.



7.4 Log & Index Evidence

Representative **log samples** and **index queries** confirm correct field parsing and ECS alignment:

- **Syslog / SSH Logs:** Parsed fields include event.outcome, source.ip, user.name.
- **Suricata Logs:** Fields such as alert.signature_id, network.transport, and geoip.* enriched for threat analysis.
- **Keycloak Logs:** JSON fields mapped to ECS for event.type, client.id, error.code.
- **Kubernetes Logs:** Proper labeling with namespace, pod.name, container.name.

7.5 Supporting Scripts & Configurations

The following scripts and config snippets were used for testing and validation:

- **SSH Failure Simulation Script**
A short Bash script using ssh in a loop to generate multiple failed logins.
- **Nmap Scan Simulation**
Used for generating Suricata IDS alerts.
- **Pod Crash Script**
A simple kubectl delete pod command to trigger Kubernetes pod monitoring rules.
- **Elastic Agent Policy YAMLS**
Tenant-specific policies for system logs, Suricata, and Kubernetes integrations.

All scripts were executed in isolated test windows to avoid cross-tenant noise.

8. Market Requirement ↔ Evidence Mapping Table

The following table provides a **clear traceability matrix** between the **functional and security requirements** for the SIEM subsystem and the **evidence collected** during implementation and validation. This ensures full visibility into how each requirement has been addressed and demonstrated through configuration, logs, dashboards, and detection scenarios.

S. No.	Requirement	Implementation Detail	Evidence Reference
1	Centralized collection of security telemetry from endpoints, network, authentication systems, and containers	Wazuh + Elastic Agent for endpoints; Suricata IDS for network; Keycloak for authentication; Elastic Agent DaemonSet for Kubernetes telemetry	Section 7.1 – Telemetry ingestion validation screenshots and logs (Wazuh, Suricata, Keycloak, K8s)
2	Multi-tenant telemetry segregation and visibility	Tenant-specific agent enrollment, index patterns, detection rules, and dashboards for Tenant 1 and Tenant 2	Section 5 – Configurations (tenant tags, index naming); Section 7.3 – Dashboard filters per tenant
3	Ingestion of Suricata IDS alerts and geo-enrichment for threat analysis	Suricata EVE JSON logs forwarded through Elastic Agent with GeoIP enrichment for Elasticsearch indexing	Section 7.1.2 – Suricata ingestion evidence; Section 7.4 – log samples with geoip fields
4	Ingestion and parsing of Keycloak authentication logs for failed login monitoring	Logstash Grok + JSON filters applied to Keycloak logs; ECS mappings for username, client_id, event type, and error	Section 5.5 – Keycloak log integration; Section 7.1.3 – parsing validation screenshots

5	Kubernetes container and pod event monitoring	Elastic Agent DaemonSet deployed in the cluster for pod lifecycle and container stdout/stderr collection	Section 5.6 – Kubernetes agent configuration; Section 7.1.4 – Kubernetes ingestion evidence
6	Detection of SSH failed logins separately for Tenant 1 and Tenant 2	Separate Kibana detection rules per tenant, scoped via index patterns and source IP/user filtering	Section 6.2 – Detection workflow; Section 7.2 – SSH rule trigger evidence
7	Detection of Keycloak authentication failures per application and per machine	Multiple detection rules created for each application under each tenant; event_type filters applied	Section 6.2 – Detection logic; Section 7.2 – Keycloak failed login scenarios
8	Detection of abnormal pod behavior in Kubernetes	Kubernetes monitoring rule created to detect unexpected pod restarts and namespace misuse	Section 6.5 – Kubernetes monitoring workflow; Section 7.2 – Pod crash simulation evidence
9	Dashboards for real-time visualization of endpoint, network, authentication, and Kubernetes events	SSH, Keycloak, Suricata, and Kubernetes dashboards configured in Kibana with tenant-based filters	Section 7.3 – Dashboard visualization evidence
10	Analyst investigation workflow for detection triage, IOC extraction, and correlation	Defined structured analyst workflow leveraging Kibana dashboards, filters, and multi-source correlation	Section 6.6 – Analyst investigation workflow
11	Evidence of end-to-end ingestion → detection → visualization pipeline	Functional telemetry flow validated from sources through Elastic Agent, Logstash, Elasticsearch,	Section 6.7 – End-to-end workflow diagram; Section 7 – Ingestion +

		detection rules, and Kibana dashboards	Detection + Dashboard evidence
12	Clear documentation of configurations, detection logic, and dashboards for audit and knowledge transfer	Complete configuration breakdown documented for Wazuh, Elastic Stack, Suricata, Keycloak, Kubernetes, and detection rules	Section 5 – Configurations & Customizations
13	Validation of SIEM operation through simulated attacks (SSH brute force, Keycloak login failures, pod anomalies)	SSH brute-force scripts, Keycloak failed login injections, Suricata alert generation (Nmap), and Kubernetes pod crash simulations executed to validate detection logic	Section 7.2 – Detection validation table

9. Conclusion

The SIEM subsystem implemented as part of the **Citadel MVP** successfully establishes a **multi-tenant, multi-source security monitoring foundation** aligned with modern SOC practices. By integrating **endpoint, network, authentication**, and **Kubernetes telemetry** into a centralized Elastic Stack backend, the environment now supports real-time security visibility, detection, and investigation for both Tenant 1 and Tenant 2.

The deployment demonstrates several key **achievements**:

- **End-to-end telemetry coverage** across Wazuh, Suricata, Keycloak, and Kubernetes components.
- **Tenant-aware ingestion and detection**, ensuring clean logical separation of data and alerts.
- **Custom detection rules** for SSH failed logins, authentication failures, and pod anomalies.
- **Operational dashboards** enabling SOC analysts to investigate incidents efficiently with real-time data.

- **Validation through realistic attack simulations**, proving detection logic effectiveness across multiple scenarios.

In addition, the environment has been built with **future extensibility** in mind. The architecture and configurations allow for:

- Integration of **additional telemetry sources**, including cloud-native logs (e.g., Azure, AWS).
- Expansion of detection logic to cover **advanced attack techniques** and cross-layer correlations.
- Implementation of **role-based access controls** and scaling Elasticsearch for production workloads.

At the same time, certain **gaps and constraints** were identified that inform the next phase of enhancement:

- Current ELK deployment is single-node, without clustering or high availability.
- Log retention and historical analysis capabilities are limited to MVP timelines.
- Multi-tenant RBAC is not yet fully implemented.

Despite these limitations, the Citadel SIEM MVP delivers a **robust and operationally effective security monitoring capability** that meets all core client requirements for the initial phase. It provides SOC teams with **real-time visibility**, **investigation tooling**, and a **clear upgrade path** toward a fully mature SIEM-SOAR ecosystem.

The next planned steps include:

- Scaling the Elastic Stack for resilience and retention.
- Expanding telemetry sources to cloud and application layers.

This phased, well-structured approach ensures that **security monitoring capabilities evolve systematically** while maintaining operational stability and audit readiness.