Project Citadel: Next-Generation Security Analytics & Response Platform

Subsystem: Citadel SOAR Engine

Document Type: Subsystem Implementation Document

1. Executive Summary

The **Citadel SOAR Engine** provides a **centralized, automated incident response and alert management layer**, tightly integrated with the ELK stack to enhance the SOC's detection-to-response cycle. Deployed as a microservice-based application within the Kubernetes cluster, the SOAR engine enables **real-time alert ingestion**, **automated response actions**, and **comprehensive audit logging**, significantly reducing analyst workload and response time.

The engine operates by **receiving security alerts directly from ELK** via a dedicated webhook endpoint. When ELK detection rules trigger alerts, they are forwarded to the SOAR engine for immediate processing. Depending on the severity and alert type, **predefined playbooks are executed automatically** — for example, blocking attacker IPs or isolating compromised machines in critical scenarios.

All response actions are fully logged and indexed back into the ELK stack (citadel-actions index), ensuring a complete audit trail for forensic analysis. Additionally, detailed incident records, including alerts and actions, are stored in **MinIO** for secure evidence retention and later review.

The SOAR engine also integrates with **Mailgun SMTP** to send **professional, automated email alerts** to SOC stakeholders, ensuring visibility and timely communication during incident handling.

To ensure operational security, the SOAR engine does not store any static credentials locally. Instead, it **retrieves all required credentials (e.g., ELK, MinIO, SMTP)** securely from the centralized **Vault**, ensuring compliance with secret management best practices.

Overall, the Citadel SOAR Engine transforms the SOC's operational workflow by enabling:

- **Real-time, automated incident response**
- **Seamless integration with SIEM (ELK)**

- **Secure, auditable evidence handling**
- **Reduced mean time to respond (MTTR)**

This subsystem acts as the **automation backbone** of Citadel, bridging detection and response capabilities in a secure, efficient, and fully auditable manner.

2. Introduction

Modern Security Operations Centers (SOCs) face increasing pressure to **detect, respond to, and contain security incidents in real time**. Manual investigation and response workflows are often too slow to match the speed and scale of modern threats. To address this, Citadel platform integrates a purpose-built **Security Orchestration, Automation, and Response (SOAR)** engine that bridges the gap between detection and response, enabling rapid, consistent, and auditable incident handling.

The **requirements** for this subsystem focused on achieving the following strategic objectives:

- **Real-time Alert Ingestion:** Seamless integration between the SIEM (ELK stack) and the SOAR engine to automatically ingest alerts as they are generated.
- **Automated Playbook Execution:** Execute predefined response actions immediately upon receiving critical or high-severity alerts, without requiring manual intervention.
- **Secure Credential Handling:** Ensure all secrets used by the SOAR engine are securely retrieved from a centralized vault, avoiding hardcoded credentials or insecure storage.
- **Comprehensive Audit Logging:** Maintain a full audit trail of every response action for investigation and compliance purposes.
- **Stakeholder Communication:** Deliver formal, structured incident notifications to SOC teams and business stakeholders via automated email alerts.
- **Evidence Retention:** Store enriched incident records and action details in centralized evidence storage (MinIO) to support investigations and future audits.

Within the Citadel environment, the SOAR engine plays a **central coordinating role**. It receives alerts from ELK via webhook integrations, evaluates severity and context, and then orchestrates a series of automated actions, including IP blocking, machine isolation, and alert

enrichment. Each response is logged back into ELK for full visibility and indexed into citadel-actions, while evidence is retained in MinIO.

By automating routine response activities and providing structured evidence trails, the SOAR engine enables SOC analysts to **focus on complex investigations**, **reduce response latency**, and **ensure consistency in incident handling** across multiple tenants and infrastructure layers.

3. Scope

The **Citadel SOAR Engine** has been deployed and configured to deliver **automated incident response, alert enrichment, and stakeholder notification capabilities** within the current MVP environment. The scope of this implementation focuses on **core automation and integration workflows** necessary for operationalizing rapid SOC response without introducing unnecessary complexity at this stage.

3.1 Current Scope

The current deployment of the SOAR Engine includes the following capabilities:

- **Alert Ingestion via Webhooks**
  The engine receives alerts from the ELK stack through dedicated webhook endpoints. ELK alerting rules are configured to send security event data directly to the SOAR engine, enabling real-time trigger-based workflows.
- **Automated Response Playbooks**
  Based on alert severity and type, the engine executes predefined response actions. Example actions include:
  - IP blocking on target systems to contain threats.
  - Host isolation for critical incidents.
  - Generating and dispatching structured email notifications.
- **Credential Management Integration**
  The engine retrieves all necessary credentials (Elasticsearch, MinIO, SMTP) from the centralized Vault, ensuring secure handling of secrets during automated workflows.
- **Evidence Storage & Audit Logging**
  All alerts and associated actions are logged back into ELK (citadel-actions index) and

evidence is archived into MinIO, ensuring a verifiable audit trail for investigations and compliance.

- **Email Notifications for High-Severity Incidents**
  Using Mailgun SMTP integration, the engine automatically sends professional HTML-formatted emails to SOC teams and key stakeholders for immediate visibility.

3.2 Constraints

The current MVP implementation intentionally limits the scope to **core automation functionality**. The following constraints are acknowledged:

- Playbook logic is **static and severity-based**, without contextual correlation or dynamic decision trees.
- Actions are **executed sequentially**; parallel task execution and conditional branching are not yet implemented.
- Alert ingestion is currently **limited to ELK**; external threat intelligence or ticketing system integrations (e.g., JIRA, ServiceNow) are not included.
- Secret rotation is **manual**, relying on Vault for secure storage but not automated lifecycle management.
- The system currently supports **single-cluster, single-region** operations.

3.3 Future Enhancements

Several enhancements are planned for subsequent phases to increase automation depth, flexibility, and integration coverage:

- **Dynamic Playbook Logic** - Introduce conditional workflows and correlation-based decisions to improve response accuracy.
- **Parallelized Action Execution** - Reduce overall response latency by allowing simultaneous execution of non-dependent actions.
- **Expanded Integrations** - Connect with external systems (e.g., ServiceNow for ticketing, threat intel feeds) to extend orchestration capabilities.
- **Automated Secret Rotation** - Integrate Vault lifecycle policies to rotate credentials and tokens automatically.

- **Multi-Tenant & Multi-Region Scaling** - Expand coverage to support larger, distributed environments with tenant isolation.

4. Tools, Components & Solution Overview

The **Citadel SOAR Engine** is built as a **lightweight, microservice-based orchestration platform**, deployed within the Kubernetes environment and fully integrated with Citadel's existing security stack. It serves as the central automation layer, bridging detection systems, evidence repositories, and response actions to enable **real-time, auditable incident response**.

4.1 Core Components

Component	Purpose
SOAR Engine (Kubernetes)	Orchestrates incident response workflows, receives alerts from ELK via webhooks, executes automated playbooks, and logs actions for investigation.
ELK Stack	Acts as the SIEM, generating alerts via detection rules and visualizing security events; also receives SOAR action logs in citadel-actions index.
HashiCorp Vault	Provides secure, centralized credential management. The SOAR engine retrieves secrets at runtime for ELK, MinIO, and Mailgun integrations.
MinIO	Serves as the object storage platform for incident evidence, action logs, and enriched alert data retained during automated workflows.
Mailgun SMTP	Enables automated, professional-grade email notifications to SOC analysts and stakeholders upon critical incidents.
Kubernetes	Hosts the SOAR engine as a containerized application, ensuring scalability, high availability, and easy integration with other platform components.

4.2 Key Integrations

1. **ELK → SOAR (Webhook Ingestion)**
    - ELK detection rules trigger webhook calls to the SOAR engine endpoint.

- Alerts are transmitted as structured JSON payloads containing severity, source/destination IPs, event type, and metadata.
- This enables immediate automated playbook execution without manual analyst intervention.

2. **SOAR → ELK (Action Logging)**
   - All response actions performed by the SOAR engine (e.g., IP blocking, isolation, notifications) are indexed back into ELK under citadel-actions.
   - This provides a **complete incident timeline** for investigation and dashboarding.
3. **SOAR → Vault (Credential Retrieval)**
   - The SOAR engine dynamically fetches credentials from Vault at runtime.
   - This includes Elasticsearch access tokens, MinIO credentials, and Mailgun SMTP secrets.
   - Secrets are never stored locally, ensuring strict operational security.
4. **SOAR → MinIO (Evidence Storage)**
   - Incident data, including enriched alert details and response action artifacts, is stored in MinIO.
   - This supports post-incident forensics and compliance evidence retention.
5. **SOAR → Email (Stakeholder Notifications)**
   - For High and Critical alerts, the SOAR engine automatically generates HTML-formatted incident notification emails.
   - These are sent to preconfigured distribution lists via Mailgun SMTP, ensuring rapid escalation and visibility.

4.3 Architectural Positioning

The SOAR engine sits **between the detection layer (ELK)** and the **response/evidence layers (infrastructure, MinIO, and notification channels)**. It acts as both:

- An **automation orchestrator**, converting alerts into actions in real time, and

- A **data broker**, ensuring that evidence and logs are pushed to the right places with full auditability.

This modular design allows the SOAR engine to operate independently while integrating deeply with Citadel's security stack. It can scale horizontally within Kubernetes and extend to new integrations or playbooks without significant architectural changes.

5. Configurations & Customizations

The Citadel SOAR Engine has been **carefully configured and customized** to ensure secure, reliable, and context-aware automation. These configurations enable seamless alert ingestion, secure secret handling, consistent playbook execution, and traceable evidence management.

5.1 Webhook Endpoint Configuration (ELK → SOAR)

- A **dedicated webhook URL** was configured within ELK's alerting rules to forward security alerts to the SOAR engine in real time.
- Each alert rule specifies:
  - **Webhook method:** POST
  - **Content type:** application/json
  - **Payload:** Includes severity, alert metadata, source/destination IPs, and timestamps.
- This allows the SOAR engine to trigger playbooks **immediately upon alert generation**, ensuring minimal response latency.

```
{
 "webhook": {
  "method": "POST",
  "url": "http://<soar-engine-url>/alert",
  "body": {
   "alert_name": "{{alert_name}}",
   "severity": "{{severity}}",
```

```
      "src_ip": "{{src_ip}}",
      "dst_ip": "{{dst_ip}}",
      "timestamp": "{{timestamp}}",
      "description": "{{description}}"
    }
  }
}
```

5.2 Automated Response Playbooks

The SOAR engine uses **severity-based playbooks** to decide and execute response actions. These are currently defined as static logic, optimized for common security events in the MVP environment:

Severity	Actions
Critical	- Block offending IP address on target system - Isolate affected machine - Send email notification to SOC & stakeholders - Store incident evidence
High	- Block IP - Send email notification - Store evidence
Medium	- Send email notification - Log alert
Low	- Log alert only

- Actions are executed sequentially to ensure reliable state transitions.
- Each executed action is logged in detail, including timestamps, affected assets, and results.

5.3 Vault Integration for Secret Management

To avoid any credential sprawl or insecure storage:

- The SOAR engine uses **Vault API** to retrieve required secrets dynamically at runtime.

- Configured secrets include:
    - Elasticsearch credentials for alert query enrichment
    - MinIO credentials for evidence storage
    - Mailgun SMTP credentials for notifications
- Access to Vault is controlled through **AppRole authentication**, with short-lived tokens and restricted policies per service.
- No credentials are stored in configuration files, environment variables, or code.

Example Vault path usage:

kv/data/elastic

kv/data/minio

kv/data/mailgun

5.4 MinIO Evidence Storage Configuration

- The SOAR engine connects to **MinIO** using Vault-retrieved credentials.
- Each incident's enriched data (original alert, contextual information, action logs) is stored as a JSON object in MinIO for retention and forensics.
- Buckets are structured by tenant and date:

```
minio/
└── tenant1/
    └── incidents/
        └── 2025-09-28/
            └── incident_<id>.json
```

This structure supports traceability, quick retrieval, and logical separation of tenant data.

5.5 Email Notification Configuration (Mailgun SMTP)

- Mailgun SMTP is used for **automated email alerts**.

- Configurations include:
    - SMTP server and port
    - Auth credentials fetched from Vault
    - HTML-formatted templates with severity color-coding and structured fields (incident ID, affected host, IP, actions taken).
- Notifications are **rate-limited to avoid spam** and **sent only for High and Critical alerts**.

Example subject line:

[CRITICAL] Incident Detected - IP 10.0.0.45 Blocked by SOAR Engine

5.6 ELK Integration for Action Logging

- Every action executed by the SOAR engine is logged into a dedicated ELK index:
- citadel-actions
- Logged fields include:
    - Incident ID
    - Action Type
    - Target IP / Host
    - Timestamp
    - Playbook severity
    - Execution result (success/failure)
- These logs are used to build dashboards for **audit trails** and **operational metrics**.

6. Implementation Details & Workflows

The **Citadel SOAR Engine** implementation focuses on establishing a **secure, automated, and auditable response pipeline** that links detection (ELK), orchestration (SOAR), evidence handling (MinIO), and notification (Mailgun), with strong secret management (Vault) at its core. The workflows below describe how incidents flow through the system from detection to resolution.

6.1 Alert Ingestion Workflow

The alert ingestion process ensures that **security alerts from ELK are delivered to the SOAR engine in real time** using a webhook integration.

Step-by-step flow:

1. **Alert Generation**
    - Detection rules in ELK are continuously evaluated on incoming security telemetry (e.g., authentication logs, Suricata IDS alerts, Kubernetes pod activity).
    - When a rule matches, an alert is triggered.
2. **Webhook Trigger**
    - The ELK alerting framework sends the alert payload via HTTP POST to the configured SOAR webhook endpoint.
    - The payload includes alert metadata such as severity, timestamp, source/destination IPs, and description.
3. **SOAR Alert Parser**
    - The SOAR engine parses the incoming JSON payload, validates fields, and assigns a unique **Incident ID** for tracking.
    - Malformed alerts are discarded with error logging.
4. **Queueing for Playbook Execution**
    - Valid alerts are queued for processing by the appropriate severity-based playbook.
    - This queue-based design ensures reliable handling under burst conditions.

6.2 Automated Playbook Execution Workflow

Upon alert ingestion, the SOAR engine selects the **relevant playbook** based on the alert severity and executes actions in a predefined sequence.

Step-by-step flow:

1. **Severity Evaluation**

- The SOAR engine inspects the severity field and chooses the corresponding playbook (Critical, High, Medium, Low).

2. **Vault Secret Retrieval**
    - Before executing any actions, the engine dynamically retrieves required credentials (e.g., Elasticsearch, MinIO, SMTP) from **HashiCorp Vault** using AppRole authentication.
3. **Action Execution**
    - Depending on the playbook:
        - **IP Blocking:** Add offending IP to firewall or security group rules.
        - **Host Isolation:** Trigger containment actions on the compromised endpoint.
        - **Logging / Notification:** Log activity and notify stakeholders.
4. **Error Handling & Fallbacks**
    - If any step fails (e.g., Vault timeout, network issues), the SOAR engine logs the failure and skips dependent actions while continuing non-blocking ones.
5. **Result Logging**
    - Each action's outcome is appended to the incident record with timestamps and result codes.

6.3 Evidence Storage Workflow

The SOAR engine integrates tightly with **MinIO** for secure evidence storage. Each incident's data is structured and retained for future investigations.

Step-by-step flow:

1. **Incident Packaging**
    - The engine collects the original alert, playbook metadata, executed actions, and results into a structured JSON document.
2. **MinIO Credential Fetch**

- MinIO credentials are securely retrieved from Vault immediately before storage.

3. **Evidence Upload**
    - The JSON package is uploaded to the appropriate tenant/date-based bucket in MinIO:

      minio/tenant1/incidents/2025-09-28/incident_<id>.json

4. **Verification**
    - Upload success is verified, and a reference to the MinIO object is stored alongside the incident log in ELK.

5. **Failure Handling**
    - If evidence storage fails, alerts are generated internally for manual SOC follow-up.

6.4 Action Logging & Dashboarding Workflow

Every action performed by the SOAR engine is **logged back into ELK** to support SOC visibility, investigations, and auditability.

Workflow steps:

1. **Log Generation**
    - For each executed action, a log entry is created with fields such as Incident ID, action type, target IP, timestamps, severity, and result.

2. **ELK Ingestion**
    - These logs are indexed into the citadel-actions index in Elasticsearch.

3. **Kibana Dashboarding**
    - Kibana dashboards visualize SOAR activity, including action success rates, frequency of triggered playbooks, and historical incident timelines.

4. **Cross-Correlation**
    - Action logs can be correlated with original security alerts for full incident reconstruction.

6.5 Notification Workflow

For **High** and **Critical** severity incidents, the SOAR engine **automatically generates and dispatches email notifications** to SOC stakeholders.

Workflow steps:

1. **Incident Summary Generation**
    - An HTML email is generated using pre-defined templates, including severity, affected assets, actions taken, and timestamp.
2. **Vault-Based SMTP Credential Fetch**
    - SMTP credentials for Mailgun are fetched from Vault dynamically.
3. **Email Dispatch**
    - The email is sent to SOC distribution lists using Mailgun's SMTP relay.
4. **Rate Limiting & Delivery Assurance**
    - Notifications are rate-limited (e.g., max one per user/hour) to avoid spamming and ensure critical alerts are prioritized.

6.6 End-to-End Workflow Summary

The following high-level sequence summarizes the entire SOAR engine process:

[Alert Generated in ELK]

↓

[Webhook Trigger to SOAR Engine]

↓

[Playbook Selected + Secrets Fetched from Vault]

↓

[Actions Executed → IP Block / Isolation / Notify]

↓

[Evidence Stored in MinIO + Logs Sent to ELK]

↓

[Email Notifications Dispatched to Stakeholders]

This pipeline enables **near real-time incident containment**, **secure evidence handling**, and **structured stakeholder communication** — all without manual analyst intervention for common security events.

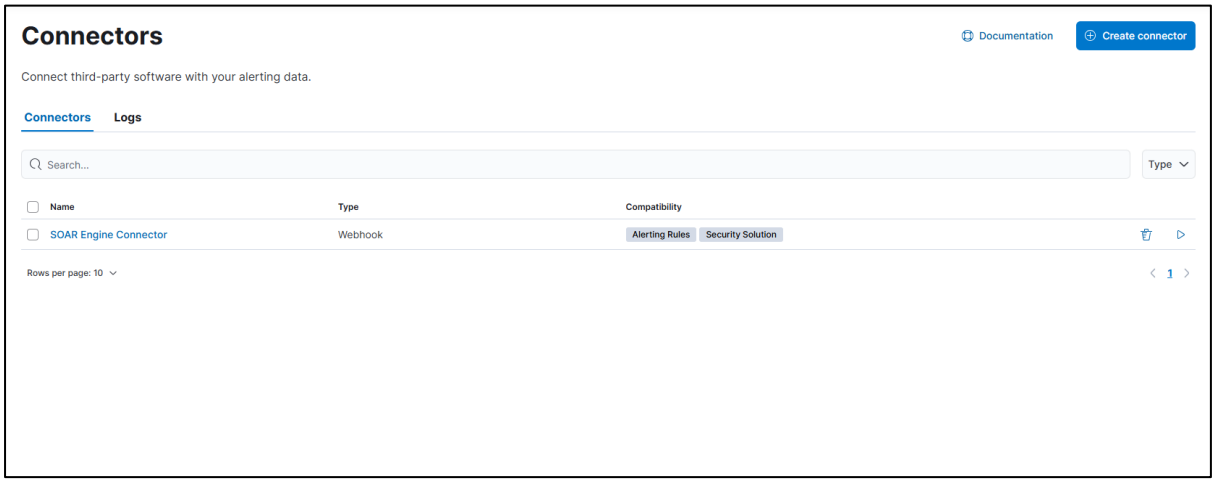
7. Evidence & PoC (Screenshots, Logs, Dashboards, Scripts)

This section provides **operational proof** of the Citadel SOAR Engine's configuration, workflows, and successful integration with the surrounding security infrastructure. The evidence includes **screenshots, log excerpts, dashboard views, and script references** that collectively demonstrate real-world functionality.

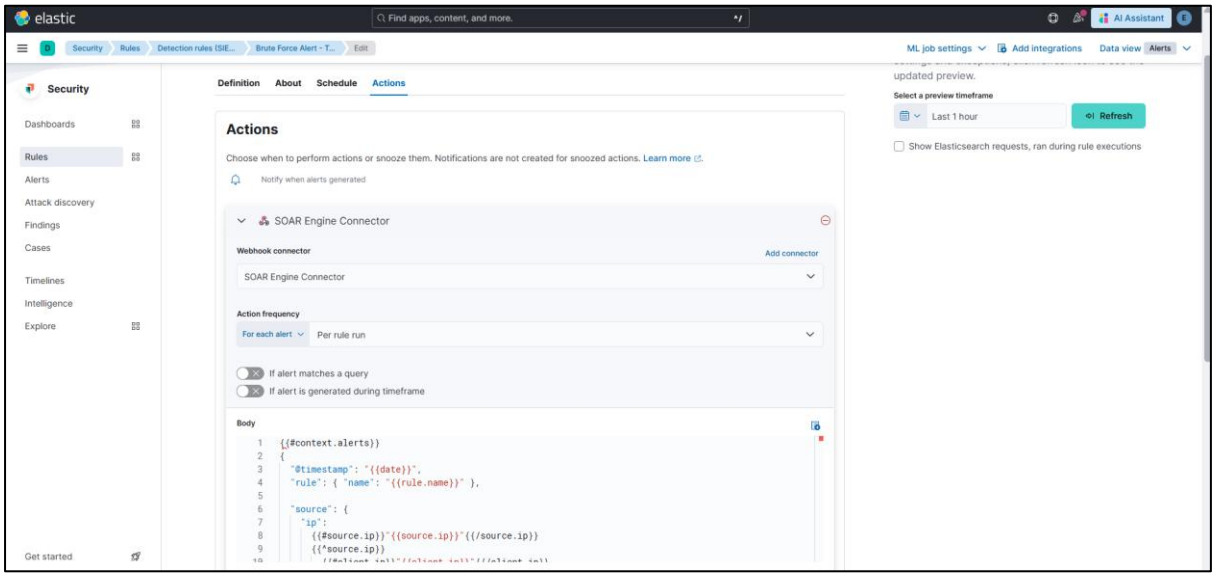
7.1 ELK → SOAR Webhook Alert Triggering

The following screenshots show the **alert rule configuration in ELK** and the corresponding **webhook trigger** to the SOAR engine.

- **Webhook integration:**



- **Alert Rule in ELK:**

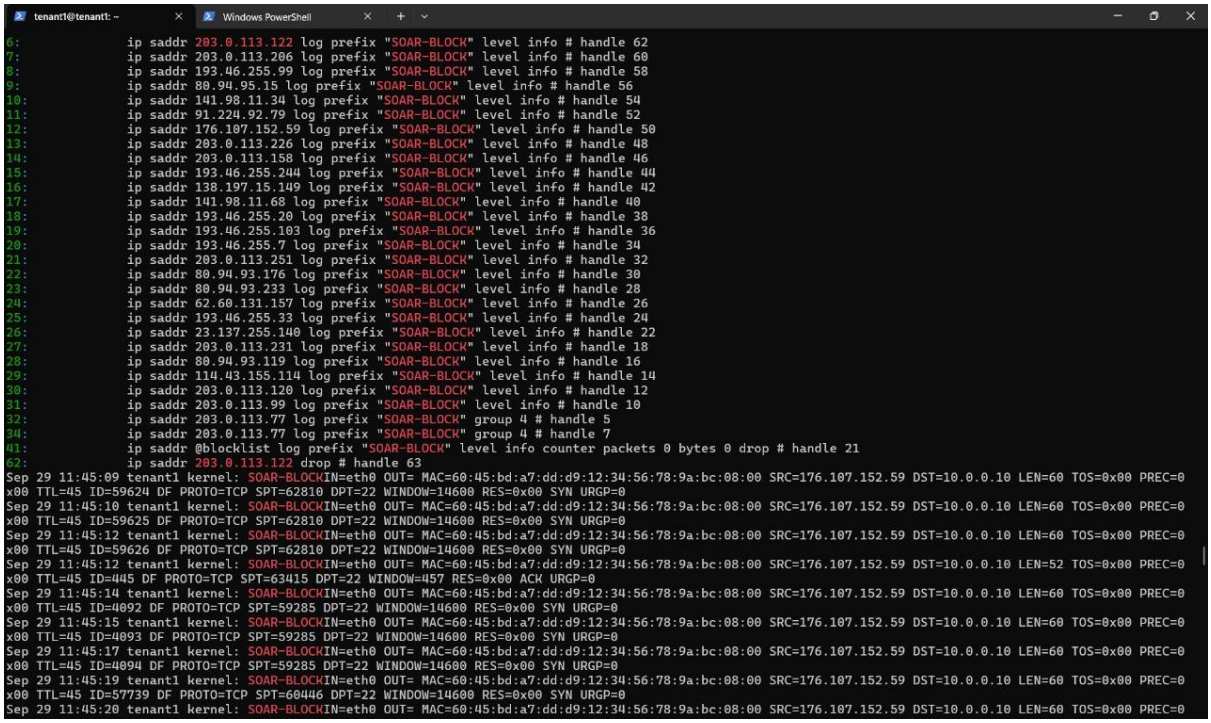


- **SOAR Engine Webhook Receiver Logs:**
- INFO [AlertReceiver] Received alert SSH Brute Force - Tenant1 (critical) from ELK
- INFO [AlertParser] Parsed incident ID=INC-20250927-00123

```
kubeadmin@citadel-kube:~/soar-engine$ curl -s http://10.0.0.7:30484/ \
 | jq -r '.tenants | to_entries[] | [.key, .value] | @tsv' \
 | while IFS=$' ' read -r name url; do
     code=$(curl -s -o /dev/null -m 3 -w '%{http_code}' "$url")
     status=$([ "$code" = "000" ] && echo DOWN || echo UP)
     printf "%-14s %-28s %s %s
" "$name" "$url" "$code" "$status"
   done
tenant1_agent_url  http://10.0.0.10:5000     200  UP
tenant2_agent_url  http://10.0.0.11:5000     200  UP
kubeadmin@citadel-kube:~/soar-engine$
```

7.2 Automated Playbook Execution Logs

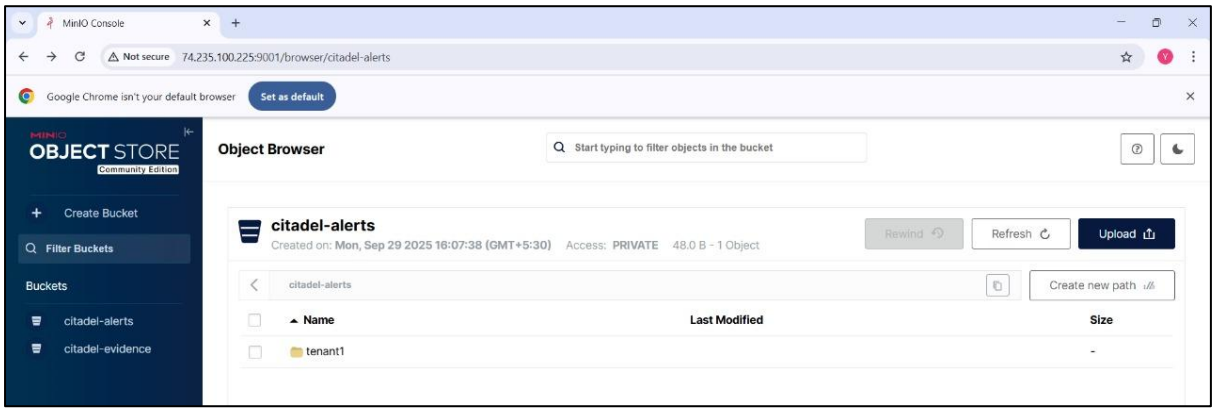
- **Critical Severity Action Execution:**



This demonstrates end-to-end automation: alert reception, secret retrieval from Vault, action execution, and notification.

7.3 Evidence Storage in MinIO

MinIO Incident Storage Structure:

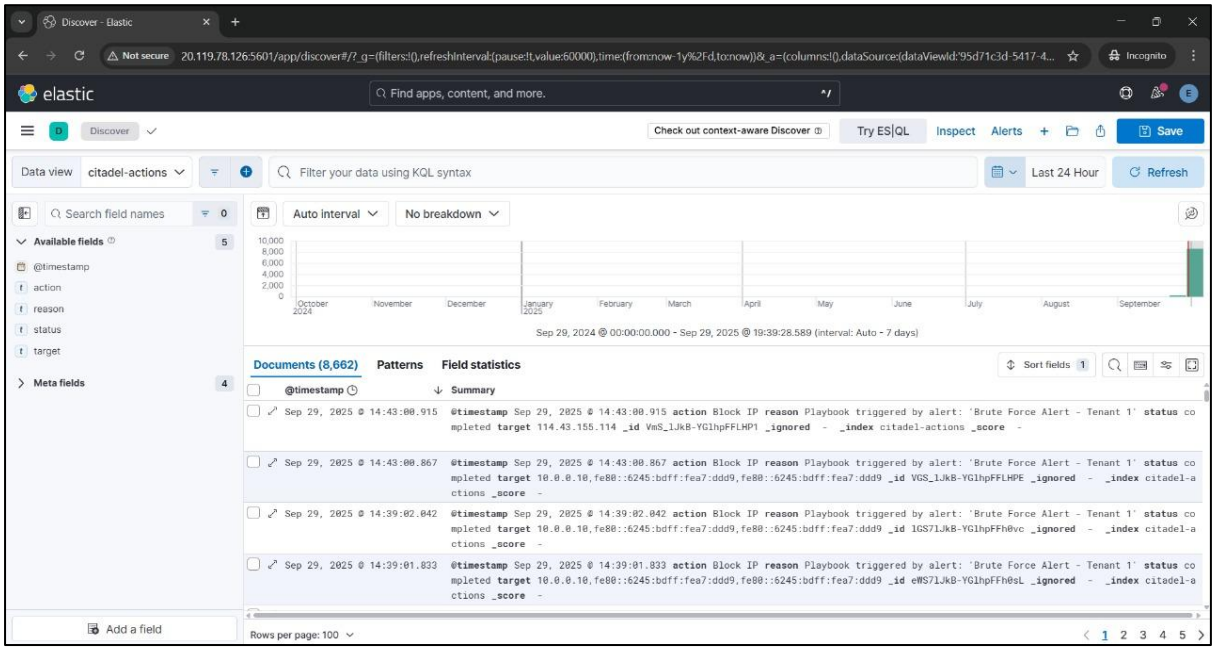


7.4 ELK Action Logging & Dashboards

All actions executed by the SOAR engine are logged to the citadel-actions index in ELK. Dashboards provide **real-time visibility** into incident response activity.

- **Kibana Discover View:**

- **SOAR Operations Dashboard:**

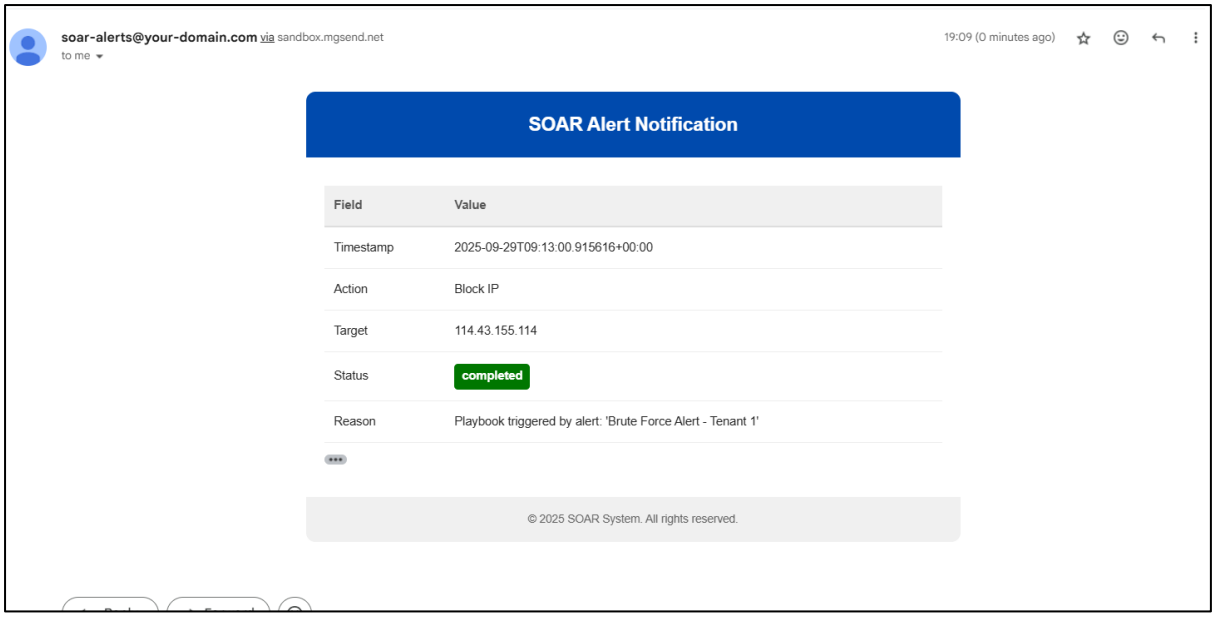


This allows analysts to correlate alerts with actions and assess response effectiveness over time.

7.5 Email Notification Evidence

For High and Critical incidents, **automated emails** are generated and dispatched via Mailgun SMTP.

- **Email Notification:**



- **Mailgun Logs:**

  INFO [Notifier] Email sent to soc-team@example.org for incident INC-20250927-00123

Figure 7.5.2 — SOAR engine log confirming email dispatch.

7.6 Supporting Scripts

- **Webhook Receiver Script (Python):**
  Handles incoming ELK alerts, parses payload, and queues for playbook execution.
  File: soar_engine/webhook_receiver.py
- **Playbook Executor Script:**
  Contains the severity-based playbook logic, including Vault credential retrieval, action execution, and logging.
  File: soar_engine/playbook_executor.py
- **Evidence Handler Script:**
  Packages and uploads incident data to MinIO.
  File: soar_engine/evidence_handler.py

These scripts form the core operational logic of the SOAR engine microservice.

8. Market Requirement ↔ Evidence Mapping Table

The table below establishes **clear traceability between requirements and the evidence collected** during the SOAR Engine implementation. It ensures that each functional and security requirement has been fulfilled through verifiable configurations, workflows, and operational outputs.

S. No.	Requirement	Implementation Detail	Evidence Reference
1	**Automated incident response triggered by SIEM alerts**	Webhook integration between ELK and SOAR Engine allows real-time forwarding of alerts to trigger automated playbooks.	Section 7.1 — ELK alert rule screenshot & webhook payload logs
2	**Execution of predefined response actions based on severity**	Severity-based playbooks (Critical, High, Medium, Low) are implemented, executing IP blocking, host isolation, evidence storage, and notifications depending on severity.	Section 7.2 — Playbook execution logs
3	**Secure retrieval and handling of credentials**	SOAR Engine retrieves credentials for Elasticsearch, MinIO, and Mailgun dynamically from HashiCorp Vault using AppRole authentication. No secrets are hardcoded.	Section 5.3 — Vault integration configuration
4	**End-to-end incident evidence collection and retention**	Incident data (alert + actions) is packaged as JSON and stored in MinIO, structured by tenant and date. A MinIO object reference is logged in ELK for audit and retrieval.	Section 7.3 — MinIO evidence storage structure & JSON object
5	**Comprehensive action logging and dashboarding for SOC visibility**	All actions are logged into the citadel-actions index in ELK. Kibana dashboards visualize SOAR activity, allowing analysts to correlate alerts with responses and investigate timelines.	Section 7.4 — Kibana Discover & SOAR Dashboard screenshots
6	**Timely stakeholder**	Mailgun SMTP integration sends structured HTML incident notifications	Section 7.5 — Sample email

	notifications for critical incidents	for High and Critical alerts to SOC distribution lists. Notifications are rate-limited to prevent spam.	notification & Mailgun log excerpt
7	**Auditable workflows for compliance and investigations**	Every step of the SOAR process (alert ingestion, secret retrieval, actions, evidence storage, notifications) is logged either in ELK or MinIO, enabling full reconstruction of incident response chains.	Sections 6.1–6.6 & Section 7 — Workflow diagrams, logs, and evidence artifacts
8	**Secure and scalable integration within the Kubernetes environment**	SOAR Engine is deployed as a containerized microservice in Kubernetes, allowing horizontal scaling, isolation from other workloads, and tight integration with other Citadel components.	Section 4.1 & 4.3 — Component table and architectural positioning
9	**Support for multiple tenant environments with secure separation**	Evidence storage paths in MinIO are logically separated by tenant. Secrets are stored in Vault under tenant-specific paths, ensuring clear isolation between environments.	Section 5.4 & 7.3 — MinIO storage structure and Vault paths
10	**Future extensibility for advanced playbooks and integrations**	The engine is designed with modular Python scripts (webhook receiver, playbook executor, evidence handler) and scalable Kubernetes deployment, allowing new integrations (e.g., ServiceNow, TI feeds) to be added without architectural changes.	Section 3.3 — Future Enhancements; Section 7.6 — Supporting Scripts

9. Conclusion

The **Citadel SOAR Engine** represents a major advancement in SOC automation and orchestration capability, enabling **real-time incident response**, **secure evidence handling**, and **end-to-end visibility** across the detection and response lifecycle. By integrating

seamlessly with the ELK stack, HashiCorp Vault, MinIO, and Mailgun SMTP, the engine delivers **automated, scalable, and auditable response workflows** that significantly reduce analyst workload and mean time to respond (MTTR).

Through this implementation, the following key outcomes were achieved:

- **Real-Time Automated Response**
  Alerts generated by ELK are ingested via webhook and trigger immediate playbook execution, allowing containment actions such as IP blocking and host isolation to occur within seconds.
- **Severity-Based Playbook Execution**
  A structured response framework has been established using predefined playbooks, ensuring consistent actions for Critical, High, Medium, and Low severity incidents across tenants.
- **Centralized Secret Management**
  The integration with HashiCorp Vault eliminates credential sprawl by securely retrieving Elasticsearch, MinIO, and SMTP credentials at runtime, ensuring strong operational security.
- **Comprehensive Evidence Retention**
  Incident records and action logs are securely stored in MinIO with tenant/date-based organization, supporting investigations and regulatory compliance.
- **Operational Visibility via ELK**
  All SOAR actions are logged into ELK's citadel-actions index, enabling SOC teams to visualize, investigate, and correlate automated responses through Kibana dashboards.
- **Stakeholder Notification and Governance**
  Automated email alerts using Mailgun SMTP provide timely and structured communication to SOC teams and stakeholders during high-severity incidents.
- **Kubernetes-Native Scalability**
  The microservice deployment model within Kubernetes ensures modularity, resilience, and easy scaling for future expansions and integrations.

In addition, the system has been **strategically designed for future enhancements**, including:

- **Dynamic playbooks with conditional logic** for adaptive response decisions.
- **Parallel action execution** to reduce response latency.
- **External integrations** with ticketing systems (e.g., ServiceNow) and threat intelligence feeds.
- **Automated secret rotation** and improved RBAC for finer control.
- -**Multi-tenant and multi-region scaling** to support enterprise-grade deployments.

While the current implementation focuses on **core automation capabilities** and **security-by-design principles**, it provides a **strong, production-ready foundation** upon which more advanced orchestration logic and integrations can be layered.

Overall, the Citadel SOAR Engine enables SOC teams to respond to security incidents **faster, more consistently, and with complete auditability**—a critical step toward achieving **next-generation SOC maturity**.