Project Citadel: Next-Generation Security Analytics & Response Platform

Subsystem: Citadel Secrets & Evidence Management subsystem

Document Type: Subsystem Implementation Document

1. Executive Summary

The **Citadel Secrets & Evidence Management subsystem** provides a **secure, centralized foundation** for handling sensitive credentials, enforcing authentication, and storing critical evidence data across the Citadel platform. This subsystem is essential for maintaining the integrity, confidentiality, and operational reliability of security workflows, particularly within multi-tenant environments.

Three core components form this architecture:

- **HashiCorp Vault** serves as the **central secret management system**, securely storing and managing lifecycle operations for credentials used by platform services, automation pipelines, and security tools. It holds sensitive data such as MinIO credentials, Elasticsearch connection details, and application-specific secrets for both tenant environments.
- **Keycloak** acts as the **centralized identity and access management platform**, ensuring strong authentication and authorization for applications and services. It manages user sessions, realm-specific retention policies, and integrates with tenant applications for secure logins.
- **MinIO** provides **object storage for evidence and operational data**, including alert and action logs generated by the Citadel SOAR and detection systems. Its storage is protected by Vault-managed credentials and governed through Keycloak-enforced access controls.

Additionally, the **SOAR Automation Team** integrates with this subsystem to securely fetch credentials from Vault, authenticate via Keycloak, and interact with MinIO and Elasticsearch during automated response workflows. This ensures that secrets never reside in code or configuration files, maintaining strict separation of duties and reducing credential exposure risks.

This tightly integrated architecture ensures that:

- All secrets are centrally managed and rotated, reducing risk of compromise.

- Authentication and authorization are standardized across services.
- Evidence is stored securely with clear access control and auditability.
- Automation workflows can securely interact with protected resources without hard-coding credentials.

Overall, the Citadel Secrets & Evidence Management subsystem represents a **mature, production-grade security backbone** that underpins both manual SOC investigations and automated SOAR responses. It aligns closely with security control objectives for **credential security**, **access governance**, and **evidence retention integrity**.

2. Introduction

2.1 Requirements & Objectives

Modern SOC environments require a **secure, centralized system** for managing credentials, enforcing authentication, and preserving operational evidence with integrity. The key requirements for this subsystem included:

- **Secure Secret Storage & Lifecycle Management**
  All sensitive credentials—including service accounts, database passwords, API keys, and platform integration tokens—must be stored in a **centralized, access-controlled vault**, with rotation and retrieval mechanisms that avoid hard-coding secrets in automation or configuration files.
- **Centralized Authentication & Access Control**
  User and service authentication must be standardized across the Citadel environment to ensure consistent policy enforcement and session governance, particularly in a **multi-tenant context**.
- **Evidence Storage & Auditability**
  All operational evidence—including alert logs, action logs, and automation artifacts—must be stored in a secure, tamper-resistant location with **controlled access and retention policies** suitable for forensic investigations and compliance audits.
- **SOAR Integration for Automation**
  Secrets and evidence management must support **automated workflows**, enabling the SOAR engine to securely retrieve credentials and interact with storage and

authentication services during automated response actions, without credential exposure.

These objectives collectively aim to **reduce operational risk**, **enforce least-privilege principles**, and **support security automation** without compromising credential hygiene or audit requirements.

2.2 Implemented Solution Overview

To address these requirements, the Citadel environment integrates **three core platforms**:

1. **HashiCorp Vault**
   Acts as the **central secret management platform**, securely storing MinIO credentials, Elasticsearch credentials (URL, username, password), and various application-specific secrets for both tenant environments. Vault enforces secret lifecycle policies and acts as the single source of truth for sensitive configuration data.
2. **Keycloak**
   Serves as the **identity and access management system**, providing unified authentication and authorization across applications and services. Keycloak enforces session and retention policies through realms and handles login for tenant applications. Authentication logs are also ingested into ELK for detection engineering and automation.
3. **MinIO**
   Provides **secure object storage** for operational data such as alert and action logs. It is integrated with Vault for credential security and with Keycloak for controlled access, ensuring evidence is stored with confidentiality and integrity guarantees.

Additionally, the **SOAR (Security Orchestration, Automation, and Response) team** consumes Vault-managed credentials to interact with MinIO, Elasticsearch, and Keycloak during automated workflows. This ensures that:

- Secrets are never embedded in code or stored insecurely.
- Authentication is consistently enforced.
- Evidence data flows remain fully auditable.

2.3 Strategic Role within Citadel

The **Secrets & Evidence Management subsystem** serves as the **security backbone** of the entire Citadel platform. It underpins:

- **Detection Engineering** – By ensuring secure access to Elasticsearch and evidence data for detection queries and enrichment.
- **SOAR Automation** – By enabling secrets-driven, authenticated actions in response workflows.
- **Access Governance** – By centralizing identity management via Keycloak.
- **Evidence Retention** – By storing forensic artifacts and operational data in MinIO with Vault-secured access.

This subsystem ensures that both **manual SOC operations** and **automated response mechanisms** operate securely, efficiently, and in compliance with SOC governance frameworks.

3. Scope

This section defines the **functional boundaries**, **current implementation coverage**, **operational constraints**, and **future enhancement areas** of the **Secrets & Evidence Management subsystem** within the Citadel environment. It clarifies how Vault, Keycloak, and MinIO are currently integrated and where planned improvements will extend capabilities in future phases.

3.1 Current Scope

The current implementation provides **secure secret management**, **centralized authentication**, and **evidence storage** for the Citadel MVP, covering the following areas:

a. Secret Management (Vault)

- **Centralized Secret Storage**: All sensitive credentials—including MinIO usernames/passwords, Elasticsearch connection details, and tenant-specific application secrets—are stored securely in HashiCorp Vault.
- **Multi-Tenant Support**: Secrets are organized logically for each tenant, ensuring separation and minimizing exposure risks.

- **SOAR Integration**: The SOAR engine fetches required secrets (e.g., Elasticsearch, MinIO) from Vault during automation workflows. Secrets are never hardcoded in playbooks or scripts.
- **Access Control**: Vault policies restrict secret retrieval based on service roles.

b. Authentication & Access Control (Keycloak)

- **Centralized Authentication**: Keycloak acts as the primary IAM layer for user and application authentication across tenants.
- **Tenant Applications**: Each tenant hosts an identical application, both of which rely on Keycloak for user login. Authentication logs are ingested into ELK for detection engineering and automation.
- **Session Management**: Keycloak realms define session retention and authentication policies, ensuring consistent access governance.

c. Evidence Storage (MinIO)

- **Secure Object Storage**: MinIO serves as the primary evidence and operational data repository, storing alert and action logs.
- **Vault-Managed Access**: MinIO credentials are stored and retrieved via Vault, ensuring that storage access is centrally secured and never exposed.
- **Integration with SOAR**: Automation workflows upload and retrieve evidence from MinIO securely during incident response operations.

d. Integration Across Components

- **Vault ↔ Keycloak**: Vault issues secrets; Keycloak governs authentication for access.
- **Vault ↔ SOAR**: SOAR playbooks securely fetch secrets at runtime.
- **MinIO ↔ Keycloak**: Keycloak governs authenticated access to MinIO buckets.
- **Logs ↔ ELK**: Keycloak login logs and evidence metadata are ingested into Elasticsearch for security monitoring.

3.2 Constraints

The following constraints apply to the current MVP deployment:

- **Manual Secret Rotation**: Vault currently holds static secrets; automated rotation and renewal workflows have not yet been implemented.
- **Basic Access Control**: Keycloak realms and roles are configured, but fine-grained role-based access control (RBAC) for MinIO buckets is limited.
- **Single MinIO Deployment**: MinIO is deployed on a single node without clustering or geo-redundancy.
- **Evidence Immutability**: Evidence storage lacks WORM (Write Once, Read Many) or cryptographic integrity validation at this stage.
- **No Automated Retention Enforcement**: Evidence retention relies on operational practices, not enforced storage lifecycle policies.

3.3 Gaps Identified

- **Automated Secret Lifecycle**
  Vault rotation and dynamic secret generation are not yet integrated with downstream services.
- **Granular Access Controls**
  Keycloak-MinIO integration currently provides coarse-grained access. Per-bucket or per-object RBAC for tenants is still pending.
- **Evidence Integrity**
  Evidence stored in MinIO is not yet protected by immutability guarantees or hash-based integrity validation.
- **Forensic Metadata**
  Metadata tagging and structured indexing of stored evidence for long-term search and retrieval are limited.

3.4 Future Enhancements

The following enhancements are planned for future phases of Citadel:

- **Dynamic Secret Generation & Rotation**
  Implement Vault-based automated credential rotation and short-lived access tokens for Elasticsearch, MinIO, and application secrets.

- **Advanced RBAC & Policy Enforcement**
  Introduce fine-grained Keycloak role mapping to MinIO buckets, with per-tenant policies and audit trails.
- **Evidence Integrity & Immutability**
  Integrate hash-chaining, WORM storage, or blockchain-backed integrity mechanisms for tamper-proof evidence retention.
- **Lifecycle Automation**
  Automate evidence retention and archival policies using MinIO bucket lifecycle rules, enforced via Vault and Keycloak.
- **Enhanced SOAR Integration**
  Extend automation playbooks to use ephemeral secrets and push validated evidence metadata directly to MinIO during response actions.

4. Tools, Components & Solution Overview

The **Secrets & Evidence Management subsystem** within Citadel integrates three core platforms — **HashiCorp Vault**, **Keycloak**, and **MinIO** — supported by SOC automation and monitoring components. Together, these form a **cohesive architecture** for secure credential storage, centralized authentication, and evidence data retention.

This section outlines the purpose, configuration role, and integration points for each component.

4.1 HashiCorp Vault – Secrets Management

Purpose:
HashiCorp Vault serves as the **authoritative store for sensitive credentials** and secrets across the Citadel environment. It ensures that passwords, tokens, and API keys are never hardcoded or stored insecurely in code, configurations, or automation scripts.

Key Functions:

- **Centralized Secret Storage:**
  Vault stores critical credentials including:
  - MinIO access credentials (username/password)
  - Elasticsearch URL and credentials (used by detection and SOAR workflows)

    - Application-specific secrets for both tenant machines

- **Role-Based Access:**
  Vault policies are used to restrict which services or automation workflows can retrieve specific secrets.
- **SOAR Integration:**
  The SOAR engine securely fetches secrets at runtime for MinIO, Elasticsearch, and other services during automated playbook execution.
- **Tenant Segregation:**
  Secrets are logically organized per tenant to maintain boundary separation and reduce risk of cross-tenant exposure.

Integration Points:

- **With SOAR:** Vault acts as the credential provider for response automation.
- **With MinIO & Elasticsearch:** Vault holds the credentials required for ingestion, query, and storage actions.
- **With Applications:** Application secrets for tenant-specific apps are stored here for consistent retrieval.

4.2 Keycloak – Authentication & Access Control

Purpose:
Keycloak provides **centralized identity and access management** for Citadel services and applications. It ensures standardized authentication across the platform and enforces realm-based session and retention policies.

Key Functions:

- **Centralized Login & SSO:**
  Keycloak manages user authentication for tenant applications (one deployed per tenant), SOC operator consoles, and administrative dashboards.
- **Session Retention & Realm Policies:**
  Keycloak realms define session expiry, retention, and authentication flows for different user groups and tenants.

- **Authentication Logs:**
  Login and session events from Keycloak are ingested into the ELK stack to support detection engineering, threat hunting, and automation triggers.
- **Access Governance:**
  Keycloak provides unified access control across multiple backend services, ensuring SOC operators, applications, and automation systems follow the same identity layer.

Integration Points:

- **With Tenant Applications:** Each tenant app uses Keycloak for user login and session management.
- **With ELK:** Authentication logs are streamed to Elasticsearch for monitoring and rule-based detections.
- **With MinIO:** Future integration for Keycloak-based access control over object storage buckets.
- **With SOAR:** Automation workflows can authenticate through Keycloak tokens where required.

4.3 MinIO – Evidence & Data Storage

Purpose:
MinIO serves as the **central object storage platform** for Citadel, used to retain **evidence artifacts**, **alert logs**, and **automation output data** in a secure and structured manner.

Key Functions:

- **Evidence Repository:**
  MinIO stores operational data such as:
  - Alert and action logs from SOAR workflows
  - Detection outputs and investigation artifacts
  - Scripts, configuration backups, or exported dashboards used as forensic evidence

- **Vault-Managed Credentials:**
  All MinIO access credentials are securely stored in Vault and retrieved dynamically by authorized services.
- **Secure API Access:**
  SOC operators and automation scripts interact with MinIO through authenticated API calls rather than static credentials.
- **Multi-Tenant Storage Strategy:**
  Data for each tenant is stored in logically separated buckets, ensuring clean evidence segregation.

Integration Points:

- **With Vault:** Credential retrieval and secret rotation (planned) for secure bucket access.
- **With SOAR:** Storage of incident evidence and response outputs.
- **With Detection Systems:** Storage of exported logs, dashboards, or artifacts from investigations.
- **With Keycloak:** Planned future integration for role-based access control.

4.4 SOAR & Automation Workflows

Purpose:
The SOAR subsystem interacts with Vault, Keycloak, and MinIO to **execute secure, credential-aware automation workflows** during incident response.

Key Functions:

- **Secret Retrieval:**
  SOAR playbooks dynamically fetch credentials from Vault at runtime.
- **Authenticated Actions:**
  Using Keycloak for app authentication and Vault for service credentials, SOAR executes secure automated actions.
- **Evidence Handling:**
  SOAR pushes investigation logs and response artifacts to MinIO for centralized storage, ensuring traceability.

- **Event Enrichment:**
  SOAR workflows may enrich alerts with data stored in MinIO or retrieved through Elasticsearch.

Integration Points:

- **Vault:** Secrets for MinIO, Elasticsearch, and service accounts.
- **MinIO:** Evidence upload and retrieval.
- **Keycloak:** Authentication for applications involved in workflows.
- **SIEM:** Enrichment and alert retrieval for trigger events.

4.5 ELK Stack (Supporting Role)

While not a core part of secrets or evidence storage, the **ELK stack** supports:

- **Monitoring authentication activity** via Keycloak log ingestion
- **Correlating evidence uploads and incident timelines**
- **Providing dashboards for SOC operators** to investigate authentication or evidence-handling anomalies

5. Configurations & Customizations

This section outlines the **key configurations** and **environment-specific customizations** implemented across **Vault**, **Keycloak**, and **MinIO** to establish a secure, operational secrets and evidence management layer. These configurations ensure **centralized secret lifecycle control**, **strong authentication**, and **secured evidence storage**, fully integrated with SOC workflows and automation pipelines.

5.1 HashiCorp Vault Configuration

Vault is configured as the **centralized secret store** for all critical credentials across the Citadel environment.

a. Secret Storage & Organization

- **Path Structure:**
  Secrets are organized in logical paths by **tenant** and **service** to maintain separation of credentials. Example:

- secret/
  - tenant1/
    - minio/
    - elastic/
    - apps/
  - tenant2/
    - minio/
    - elastic/
    - apps/

- **Stored Secrets Include:**
  - MinIO credentials (username, password)
  - Elasticsearch URL and credentials
  - Tenant-specific application credentials (e.g., Keycloak client secrets, API tokens)
  - SOAR service integration tokens

b. Access Policies

- Vault policies are defined to ensure that only authorized services (e.g., SOAR engine, detection services) can access relevant secrets.
- Example:
  - policy-soar-minio: grants read-only access to secret/*/minio.
  - policy-soar-elastic: grants read-only access to secret/*/elastic.
- Policies are bound to Vault tokens or AppRoles used by automation scripts and services.

c. SOAR Integration

- SOAR playbooks authenticate to Vault using a dedicated AppRole.
- Secrets are fetched **at runtime** (not stored in playbook variables).
- Example workflow:

1. SOAR authenticates to Vault with its AppRole ID and Secret ID.
2. Vault returns a scoped token.
3. SOAR uses this token to retrieve MinIO and Elasticsearch credentials securely.
4. Token is short-lived to minimize exposure.

d. Security Controls

- Vault's audit logging is enabled to track all secret retrieval operations.
- TLS is enforced on all Vault communication.
- Secrets are never exposed in logs or configuration files.

5.2 Keycloak Configuration

Keycloak serves as the **central IAM** for tenant applications and system authentication.

a. Realm & Application Setup

- **Separate Realms:**
  Each tenant application is integrated under its respective Keycloak realm, ensuring isolated authentication contexts.
- **Tenant Applications:**
  Both Tenant 1 and Tenant 2 have identical applications configured to rely on Keycloak for login and session management.

b. Authentication & Session Policies

- Realms define retention and session timeout policies to enforce secure session lifecycles.
- Example:
  - Session Timeout: 30 minutes idle, 8 hours max lifespan
  - Forced re-login after session expiration
- Login flows enforce password policies and optional MFA (planned for future phase).

c. Log Integration

- Keycloak login and session events are **forwarded to Elasticsearch** for detection engineering and monitoring.
- Elastic Agent is configured to collect Keycloak server logs and parse relevant fields:
    - event.type, user.name, client.id, error.code, realm
- These logs are used to detect anomalous login behavior, session hijacking, or brute-force attempts.

5.3 MinIO Configuration

MinIO is deployed to serve as **evidence storage** and object repository for operational and security data.

a. Bucket Structure

Buckets are created to **logically segregate evidence per tenant and function**. Example:

- minio/
- tenant1/
    - alerts/
    - actions/
    - evidence/
- tenant2/
    - alerts/
    - actions/
    - evidence/
- This ensures that evidence artifacts are clearly separated between tenants, supporting investigation and compliance.

b. Access Control

- **Vault-Managed Credentials:**
  MinIO credentials are **not hardcoded** anywhere. Instead:
    - Stored securely in Vault
    - Retrieved by SOAR or authorized users at runtime

- **Access Modes:**
    - SOC operators use MinIO API/console with Vault-issued credentials.
    - Automation workflows use Vault integration to upload or retrieve evidence.

c. Evidence Storage

- **Evidence Types Stored:**
    - SOAR automation logs (alert actions, response results)
    - Investigation exports (e.g., detection dashboards, query outputs)
    - Scripts and configuration snapshots relevant to incidents
- **Retention:**
    - Currently governed by operational policy; automated lifecycle rules planned for future.
- **Access Auditing:**
    - MinIO server logs and Vault audit logs together provide traceability for who accessed which evidence.

5.4 SOAR Automation Customizations

The SOAR engine is configured to **leverage Vault and MinIO** seamlessly during response workflows.

a. Secrets Handling

- Playbooks dynamically retrieve Elasticsearch and MinIO credentials from Vault at runtime.
- Secrets are cached in memory temporarily and invalidated after task execution.
- This eliminates static credentials in any pipeline or script.

b. Evidence Upload

- Upon incident response execution, playbooks upload structured logs, outputs, or supporting artifacts to MinIO:
    - e.g., JSON alert data, extracted indicators, enrichment results

- Uploaded objects follow naming conventions tied to incident IDs for easy retrieval.

c. Authentication

- SOAR uses Keycloak for application authentication where applicable, ensuring consistency across automation and manual operations.

5.5 Security & Audit Enhancements

Several security and observability enhancements were implemented to strengthen the subsystem:

- **Vault Audit Logs:**
  Enabled and monitored to track all secret access attempts.
- **TLS Everywhere:**
  Vault, MinIO, and Keycloak communications are protected by TLS, ensuring confidentiality of secrets in transit.
- **Indexing of Auth & Evidence Logs:**
  Keycloak logs and evidence metadata are indexed in Elasticsearch for SOC dashboards and investigations.
- **Tenant Logical Segregation:**
  All secrets and evidence are separated by tenant paths and bucket structures, reducing risk of cross-tenant leakage.

6. Implementation Details & Workflows

This section describes the **end-to-end implementation** and operational workflows of the **Secrets & Evidence Management subsystem** within Citadel. The workflows cover **secret creation and retrieval**, **authentication flows**, **evidence storage and access**, and **SOAR automation integration**, ensuring that all secrets and evidence data are handled securely, consistently, and auditable across the environment.

6.1 Secret Creation & Storage Workflow (Vault)

All sensitive credentials are created and stored centrally in **HashiCorp Vault**, ensuring they are never embedded in application code, configuration files, or SOAR playbooks.

Step-by-Step Workflow:

1. **Secret Generation**
    - Service credentials (e.g., MinIO access keys, Elasticsearch credentials, Keycloak client secrets) are generated during service onboarding.
    - Static credentials are stored initially; future phases will move to dynamic secret generation.
2. **Vault Storage**
    - Secrets are stored under structured paths:
        - secret/tenant1/minio/
        - secret/tenant1/elastic/
        - secret/tenant1/apps/
        - secret/tenant2/minio/
        - secret/tenant2/elastic/
        - secret/tenant2/apps/
    - Each secret is labeled with metadata (service, tenant, creation date).
3. **Policy Enforcement**
    - Vault policies control which AppRoles or tokens can retrieve which secrets.
    - For example, SOAR engine can only read minio and elastic secrets for the relevant tenant.
4. **Audit Logging**
    - All secret creation and retrieval operations are logged by Vault's audit backend for traceability.

6.2 Secret Retrieval Workflow (SOAR & Services)

Secrets are retrieved **dynamically at runtime** by authorized systems such as SOAR playbooks and detection services.

Flow:

1. **Authentication**

- The service (e.g., SOAR) authenticates to Vault using its AppRole credentials.
- Vault returns a scoped, short-lived token.

2. **Secret Fetch**
    - The service uses this token to retrieve the required secret (e.g., MinIO or Elasticsearch credentials) from Vault.
3. **Usage & Expiry**
    - The secret is used immediately for the intended operation (e.g., pushing evidence to MinIO, querying Elasticsearch).
    - Tokens expire quickly, ensuring credentials are not persistently stored.
4. **Audit Trail**
    - Every retrieval is logged in Vault's audit logs, capturing the role, path, and timestamp.

Example:

- A SOAR playbook triggers an incident response action.
- It authenticates to Vault → retrieves Elasticsearch URL & credentials → queries logs for enrichment → uploads results to MinIO.

6.3 Authentication Workflow (Keycloak)

Keycloak manages user and application authentication for both tenant apps and SOC services.

Workflow:

1. **Tenant Application Login**
    - A user accesses a tenant application.
    - Keycloak prompts for credentials, authenticates, and issues session tokens based on the tenant's realm policies.
2. **Session Handling**
    - Realms enforce session timeouts (e.g., 30 min idle, 8 hr max).

- Session activity and authentication events are logged.

3. **Log Ingestion**
    - Keycloak authentication logs are collected by Elastic Agent and forwarded to ELK for monitoring, detection, and alerting.
    - SOC analysts can detect anomalies such as brute-force attempts or suspicious session behavior.
4. **SOAR Use**
    - SOAR workflows can use Keycloak-issued tokens to access protected apps where needed.

6.4 Evidence Storage Workflow (MinIO)

All security and operational evidence is stored in **MinIO** object storage.
This includes **SOAR output logs**, **alert/action data**, **investigation exports**, and other artifacts.

Flow:

1. **Evidence Generation**
    - Evidence is generated by SOAR workflows (e.g., alert enrichment results), analysts (e.g., dashboard exports), or detection systems.
2. **Credential Retrieval**
    - The uploading service fetches MinIO credentials from Vault at runtime to avoid static access keys.
3. **Upload to MinIO**
    - Evidence is uploaded to tenant-specific buckets:
    - minio/tenant1/evidence/
    - minio/tenant2/evidence/
    - Naming conventions follow incidentID_timestamp_type.json for traceability.
4. **Access Control**

- Access to evidence is governed by MinIO credentials (Vault-managed) and planned Keycloak-based RBAC.

5. **Retention & Logs**
    - Evidence retention is currently governed manually; future phases will introduce automated lifecycle rules.
    - All access is logged in MinIO server logs and Vault audit logs.

6.5 Integrated SOAR Workflow

The **SOAR engine** is tightly integrated with Vault, Keycloak, and MinIO to perform secure and automated incident response actions.

Workflow Example:

1. **Incident Trigger**
    - A detection rule fires in SIEM, triggering a SOAR playbook.
2. **Vault Authentication & Secret Retrieval**
    - The playbook authenticates to Vault and retrieves required credentials:
        - Elasticsearch URL & credentials to pull logs
        - MinIO credentials to store response outputs
3. **Log Enrichment & Analysis**
    - SOAR queries Elasticsearch using the retrieved credentials.
    - Additional enrichment is performed as needed.
4. **Evidence Storage**
    - Playbook uploads structured evidence (alert context, enrichment outputs, actions taken) to MinIO.
5. **Authentication Validation (Optional)**
    - If workflow involves interacting with tenant apps, Keycloak tokens are used for secure authentication.
6. **Audit Trail**

- Every secret retrieval, evidence upload, and automated action is logged, ensuring traceability.

6.6 SOC Analyst Evidence Access Workflow

SOC analysts can retrieve stored evidence from MinIO as part of investigations.

1. **Analyst Authentication**
   - Analysts authenticate using Vault-issued credentials (or Keycloak RBAC in future).
2. **Evidence Lookup**
   - Evidence is searched by incident ID, timestamp, or type in MinIO buckets.
3. **Download & Review**
   - Artifacts are downloaded and reviewed in investigation tools or dashboards.
4. **Audit & Retention**
   - All access is logged for forensic and compliance purposes.

6.7 End-to-End Workflow Summary

[Secret Generation] → [Vault Storage]

↓

[SOAR / Services Runtime Auth → Vault Secret Retrieval]

↓

[Authenticated Operations via Keycloak]

↓

[Evidence Upload to MinIO with Vault-issued creds]

↓

[Evidence Retention & SOC Access]

↓

[Vault & MinIO Audit Logs → ELK Dashboards]

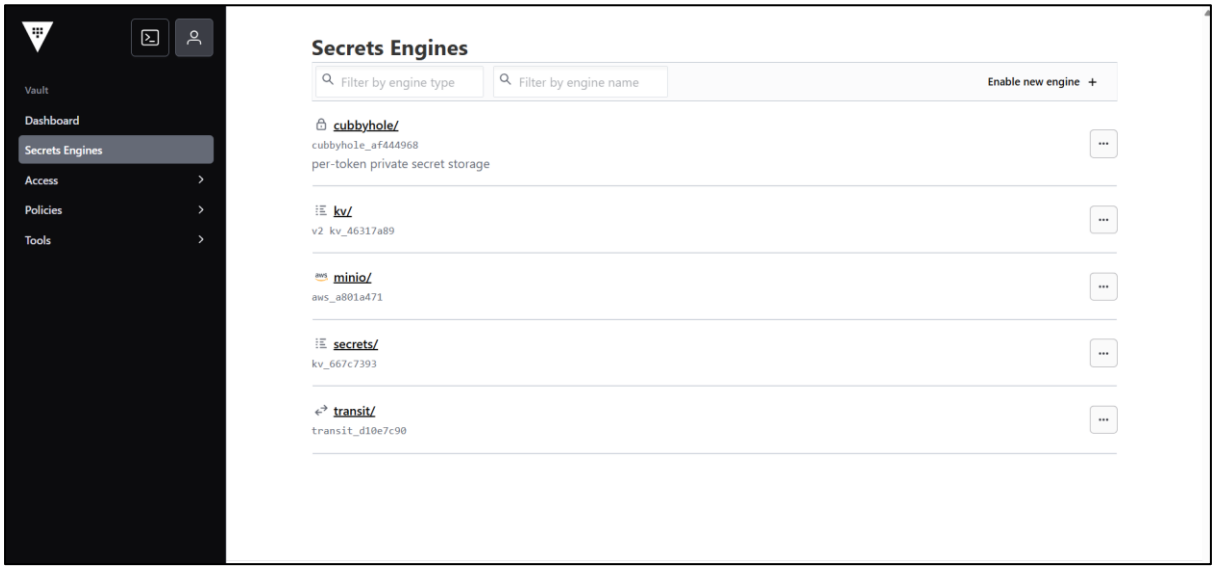
7. Evidence & PoC (Screenshots, Logs, Dashboards, Scripts)

This section presents **supporting evidence** validating the correct configuration and operation of the **Secrets & Evidence Management subsystem** across Vault, Keycloak, and MinIO. The evidence includes **screenshots, configuration extracts, logs, and workflow outputs**, collected during deployment and operational testing of the Citadel environment.

7.1 Vault – Secret Storage & Retrieval

7.1.1 Secret Storage Validation

Screenshots show **Vault UI** with structured secret paths for both tenants:



- Secrets visible in Vault include MinIO credentials, Elasticsearch credentials, and application-specific secrets.

Evidence:

- Screenshot of Vault web UI showing secret/tenant1/minio path populated
- Sample CLI output from vault kv list secret/tenant1 showing stored keys
- Policy snippet showing restricted SOAR read access to relevant paths

7.1.2 Secret Retrieval Validation

- Runtime retrieval was tested using the **SOAR AppRole**:

- SOAR authenticated using its role_id and secret_id
- A scoped token was issued and used to retrieve MinIO credentials dynamically

- Vault audit logs confirm retrieval activity with timestamp, token, and path.

Evidence:

- CLI example of vault read secret/tenant1/minio executed under SOAR AppRole
- Vault audit log extract:

{"time":"2025-09-20T10:32:18Z","type":"response","auth":{"display_name":"approle-soar"},"request":{"path":"secret/data/tenant1/minio"}}

- SOAR log snippet showing secrets fetched at runtime, not stored in playbook

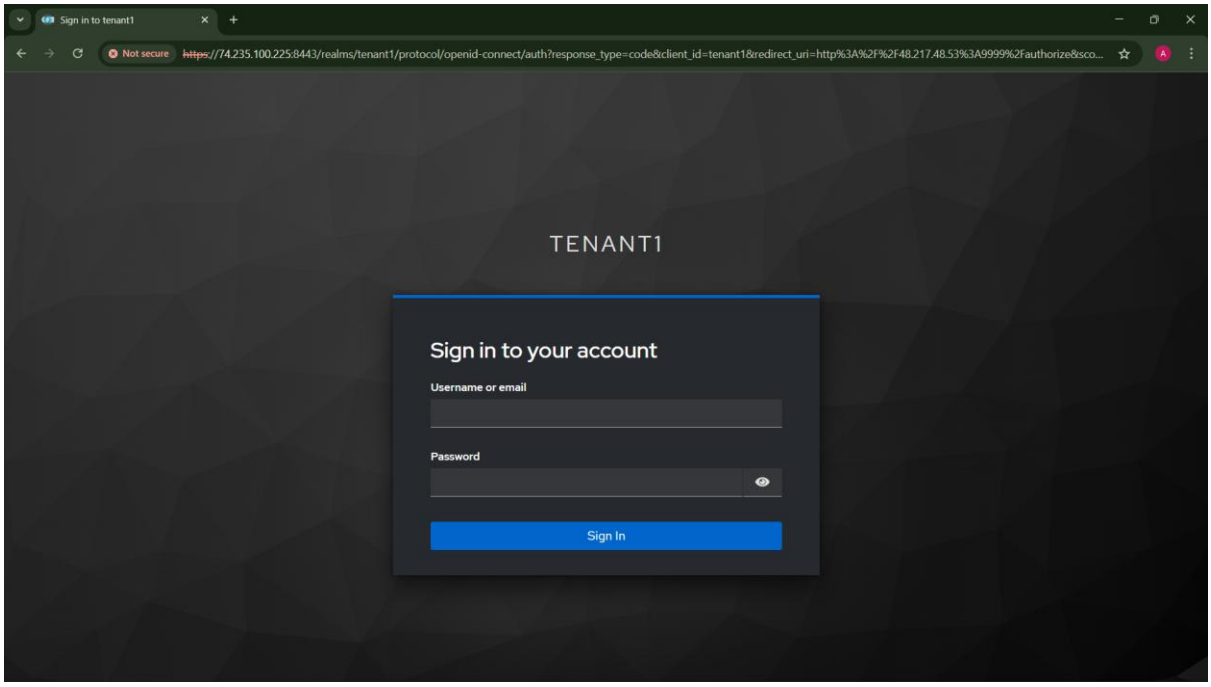
7.2 Keycloak – Authentication & Log Forwarding

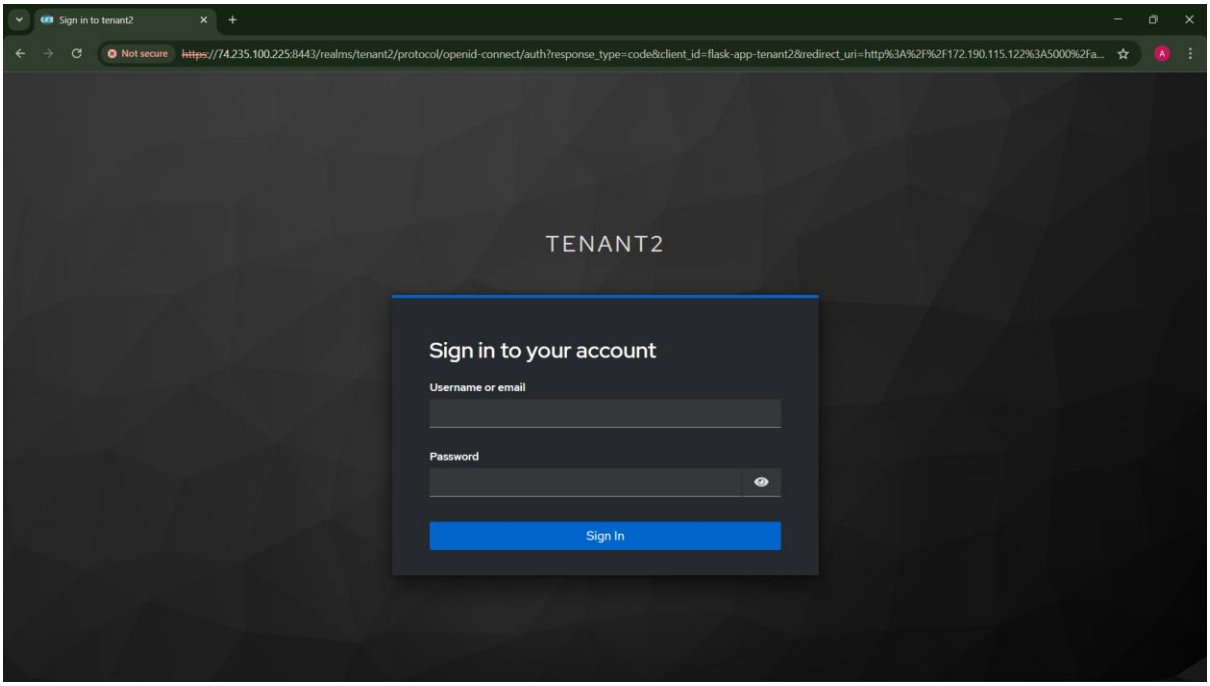
7.2.1 Tenant App Authentication

- Each tenant hosts an application authenticated through Keycloak.
- Login flows were tested using tenant-specific users, verifying realm isolation and session issuance.

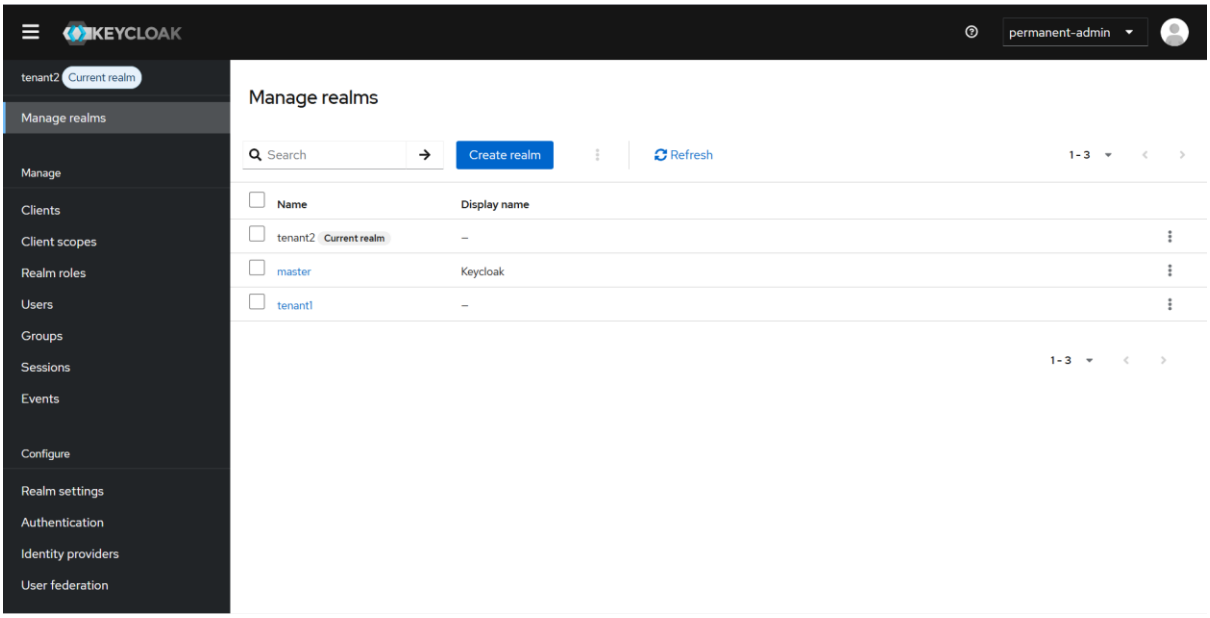
Evidence:

- Screenshot of Keycloak login screen for Tenant 1 application





- Screenshot of Keycloak admin console showing two realms: tenant1 and tenant2

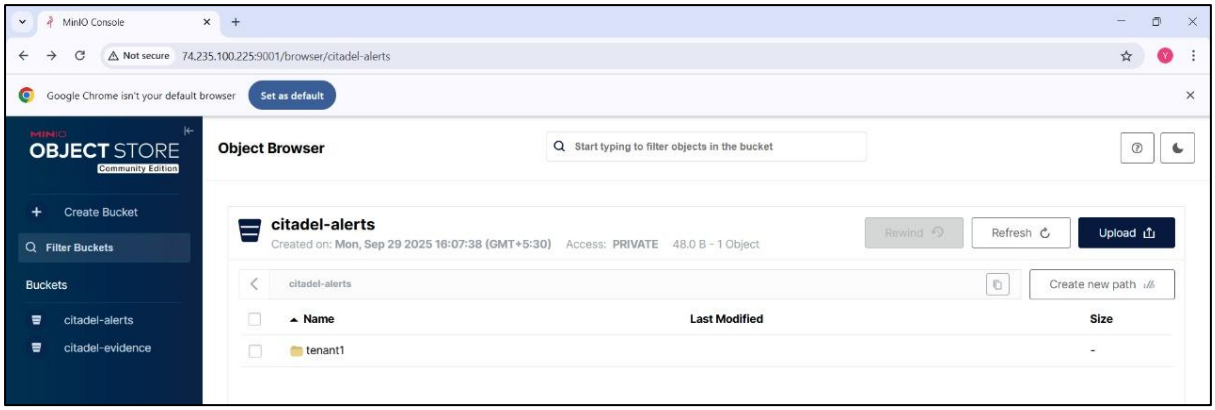


- Session details showing correct timeout and token issuance

7.3 MinIO – Evidence Storage & Access

7.3.1 Bucket Structure Validation

Buckets created per tenant and function were verified via MinIO Console:



7.3.2 Evidence Upload & Retrieval

- SOAR playbooks were tested to upload structured evidence (alert/action logs) to MinIO.
- Evidence filenames follow the pattern: incidentID_timestamp_type.json (e.g., INC102_2025-09-25T1030_alert.json)
- Retrieval was tested by analysts using Vault-issued credentials.

Evidence:

- SOAR log output showing successful evidence upload to MinIO
- MinIO server log showing PUT operation with tenant bucket and object name
- CLI retrieval example using Vault-fetched credentials:

  mc cp minio/tenant2/evidence/INC102_2025-09-25T1030_alert.json
- Corresponding Vault audit log showing credential retrieval for the analyst

7.4 End-to-End SOAR Workflow Validation

An end-to-end SOAR response workflow was executed to validate integration between Vault, Elasticsearch, and MinIO.

Scenario:

1. A detection rule triggered in SIEM for SSH brute force.
2. SOAR playbook was launched.
3. SOAR fetched Elasticsearch credentials from Vault → queried for related events.

4. SOAR fetched MinIO credentials from Vault → uploaded enrichment output to MinIO evidence bucket.
5. All actions were logged and verifiable.

Evidence:

- SOAR run log showing dynamic secret retrieval from Vault
- Elasticsearch query result snippet for enrichment data
- MinIO object listing showing newly uploaded evidence file
- Vault audit logs confirming both secret retrieval events
- Timeline diagram demonstrating flow:

  [SIEM Alert] → [SOAR Playbook] → [Vault Secrets] → [Elasticsearch Query] → [MinIO Evidence Upload]

7.5 Security & Audit Logging Evidence

To ensure operational integrity and compliance:

- **Vault Audit Logs** were reviewed to confirm traceability of all secret read operations.
- **MinIO Server Logs** captured object upload and download actions with user identifiers.
- **Keycloak Logs** captured login attempts and authentication anomalies.
- All logs were ingested into ELK for visualization and investigation.

Evidence:

- Vault audit log entries for SOAR and analyst retrieval
- MinIO server log entries showing object access
- Kibana dashboard screenshot correlating Keycloak login anomalies with MinIO evidence access events

8. Market Requirement ↔ Evidence Mapping Table

The table below provides a **clear traceability matrix** between the **functional and security requirements** for the Secrets & Evidence Management subsystem and the **evidence**

collected during the Citadel MVP implementation. This ensures visibility into how each requirement has been fulfilled through actual configurations, workflows, and operational proof.

S. No.	Requirement	Implementation Detail	Evidence Reference
1	Centralized and secure storage of sensitive credentials	All critical credentials (MinIO, Elasticsearch, tenant application secrets) are stored in HashiCorp Vault, structured by tenant and service, with strict access policies.	Section 7.1.1 – Vault secret storage screenshots & CLI output
2	Controlled retrieval of secrets by automation systems	SOAR playbooks use Vault AppRole authentication and fetch secrets dynamically at runtime; tokens are short-lived and retrievals are fully audited.	Section 7.1.2 – Vault retrieval logs, SOAR runtime output
3	Centralized authentication and access management	Keycloak manages authentication for tenant applications, enforcing realm-based policies and session retention. Authentication logs are ingested into ELK for monitoring and detections.	Section 7.2.1 – Keycloak login screenshots; Section 7.2.2 – Kibana Discover for Keycloak logs
4	Authentication activity monitoring	Keycloak login/session logs are collected via Elastic Agent, parsed into structured fields, and visualized in Kibana dashboards for anomaly detection.	Section 7.2.2 – Parsed Keycloak logs in Elasticsearch
5	Secure, centralized storage of operational evidence	MinIO deployed with per-tenant bucket structure for alerts, actions, and evidence data. Vault-managed credentials are required for access, ensuring centralized and secured evidence storage.	Section 7.3.1 – MinIO bucket structure screenshots

6	Evidence upload and retrieval through authenticated workflows	SOAR workflows fetch MinIO credentials from Vault at runtime for uploads; SOC analysts retrieve evidence using Vault-issued credentials. All access operations are logged.	Section 7.3.2 – SOAR upload logs, MinIO server logs, CLI retrieval example
7	Traceability and auditability of all secret and evidence access actions	Vault audit logs capture secret access; MinIO server logs record object operations; Keycloak logs track authentication events. All are ingested into ELK for investigation and correlation.	Section 7.5 – Audit log samples from Vault, MinIO, and Keycloak
8	Integration of secrets and evidence management into SOAR workflows	SOAR playbooks integrate Vault for secret retrieval and MinIO for evidence storage during incident response, ensuring credentials are never hardcoded and evidence is stored securely.	Section 7.4 – End-to-end SOAR workflow validation
9	Secure handling of secrets and evidence across multiple tenants	Vault secrets and MinIO buckets are logically separated by tenant. Keycloak uses separate realms for tenant apps. This ensures clean operational boundaries and reduces cross-tenant exposure risks.	Section 5.1 & 5.3 – Tenant-based Vault path and MinIO bucket structures; Section 7.3.1 evidence
10	Support for future enhancements such as automated secret rotation and evidence immutability	Architecture allows introduction of Vault dynamic secrets, lifecycle automation, and MinIO WORM or integrity mechanisms in future phases without structural redesign.	Section 3.4 – Future Enhancements

9. Conclusion

The **Secrets & Evidence Management subsystem** implemented as part of the Citadel MVP provides a **robust, centralized, and secure foundation** for handling sensitive credentials, enforcing authentication, and retaining operational evidence. By integrating **HashiCorp Vault**, **Keycloak**, and **MinIO** into a cohesive architecture, the subsystem establishes the critical security backbone required for modern SOC operations, secure automation, and compliance assurance.

The deployment has successfully achieved the following key outcomes:

- **Centralized Secret Management**
  All critical credentials are now stored and managed through Vault, eliminating hard-coded secrets, reducing exposure risk, and enabling policy-driven access control across tenants and services.
- **Unified Authentication & Governance**
  Keycloak provides centralized authentication for tenant applications and SOC components, ensuring consistent session policies and providing structured authentication logs for monitoring and detection.
- **Secure Evidence Storage & Access**
  MinIO serves as the central evidence repository, with per-tenant bucket structures and Vault-managed credentials ensuring secure storage and controlled access to investigation artifacts, logs, and automation outputs.
- **SOAR Workflow Integration**
  Secrets and evidence handling have been fully integrated into SOAR playbooks, enabling automated response workflows to operate securely without compromising credential hygiene or evidence integrity.
- **Comprehensive Audit Logging**
  Vault, Keycloak, and MinIO all produce structured logs that are ingested into ELK, allowing full traceability of secret access, authentication events, and evidence operations for security investigations and compliance audits.

In addition to these achievements, the subsystem has been **architected for future expansion** to support advanced capabilities, including:

- **Dynamic secret generation and automated rotation** via Vault for services like Elasticsearch and MinIO.
- **Fine-grained RBAC** for MinIO using Keycloak identity mapping.
- **Evidence immutability mechanisms** (e.g., WORM storage or cryptographic hash chains) to enhance forensic integrity.
- **Automated evidence retention policies** aligned with regulatory and operational requirements.

While the current MVP implementation relies on **manual secret rotation**, **basic RBAC**, and **single-node deployments**, these are deliberate design trade-offs to establish a stable and auditable baseline before introducing more complex features in subsequent phases.

Overall, the Citadel Secrets & Evidence Management subsystem delivers a **mature, production-ready security layer** that significantly strengthens SOC operations by ensuring that credentials, authentication, and evidence are managed **centrally, securely, and with full auditability**. This foundation enables SOC teams to scale automation, enhance detection capabilities, and meet compliance obligations with confidence.