Software Design Document

SDD

Project Name: **Automated Misconfiguration & Threat Detection in Public Cloud Storage**

Project Developers: **Jamal Tannous**

Asya Persan

Dan Ydov

Content

1. Introduction

a. System Overview

The system is an automated security analysis tool that identifies misconfigurations and indicators of suspicious activity in public cloud storage platforms such as AWS S3, Azure Blob Storage, and Google Cloud Storage.
It retrieves storage configuration data via cloud provider APIs, evaluates configurations using predefined rule-based checks, and analyzes synthetic access logs using unsupervised machine-learning techniques.
Detected issues include public data exposure, overly permissive access policies, missing or incorrect encryption, disabled logging, and abnormal access behavior.
All findings are consolidated and assigned severity levels (Low, Medium, High) before being presented in human-readable reports.

b. Purpose

The purpose of the system is to reduce the risk of data leakage and security incidents caused by misconfigured public cloud storage resources.
By enabling early detection of configuration weaknesses and indicators of suspicious access behavior, the system supports organizations in improving their cloud security posture and meeting compliance requirements.
The system provides automated analysis results and clear remediation recommendations to assist security and operational teams in prioritizing and addressing identified risks without modifying cloud resources directly.

c. Scope

The scope of the system includes automated analysis of public cloud storage services to identify security misconfigurations and indicators of suspicious activity. The system supports storage resources in major public cloud platforms, including AWS S3, Azure Blob Storage, and Google Cloud Storage.

Within the scope are:

- Retrieval of cloud storage configuration data using read-only access.
- Rule-based evaluation of configurations to detect common misconfigurations such as public access, weak or overly permissive permissions, missing or incorrect encryption, and disabled logging.
- Analysis of **synthetic access logs** using unsupervised machine-learning techniques to identify anomalous access patterns.
- Classification of findings into severity levels (Low, Medium, High).
- Presentation of findings through a simple CLI or minimal web interface.
- Export of scan results in human-readable and structured formats.

Out of the scope are:

- Automatic remediation or modification of cloud storage configurations.
- Analysis of non-storage cloud services (e.g., compute, databases, networking).
- Processing of real production access logs.
- Enforcement of compliance policies or security controls.

d. Constraints

The system design and implementation are subject to the following constraints:

Security Constraints

- The system shall use **read-only credentials** when accessing cloud provider APIs.
- Raw cloud configuration data and access credentials shall be kept **in memory only** and shall not be persisted to disk.
- All communication with cloud providers shall be performed using **secure protocols (HTTPS/TLS)**.
- The system shall not modify cloud storage configurations or perform automated remediation actions.

Data Constraints

- Machine-learning analysis shall be performed **only on synthetic access logs** generated for evaluation purposes.
- Processed scan results and findings may be stored locally, provided they do not contain sensitive configuration data or credentials.

Platform Constraints

- The system shall be executable on common operating systems, including **Linux, macOS, and Windows**.
- The solution shall rely on commonly available libraries and frameworks suitable for an academic project environment.

Project Constraints

- The system is developed as an **academic project** with a limited timeframe and team size.
- The implementation prioritizes clarity, modularity, and correctness over production-scale optimization.
- External dependencies on live cloud environments should be minimized where possible through the use of synthetic data and controlled test accounts.

2. System Architecture

a. Architectural Description and Design: Roles, Activities and Data

Roles

The system supports the following roles, each interacting with the system in a different operational context:

- **Security Engineer -** Primary user responsible for running security scans, reviewing detected misconfigurations and anomalies, and prioritizing remediation actions.
- **DevOps / Cloud Engineer -** Uses the system during development or deployment phases to validate cloud storage configurations and detect security risks before release.
- **Compliance Officer / Project Manager -** Reviews summarized scan results and exported reports to support compliance audits and risk assessment.

These roles interact with the system through a unified interface and do not require direct access to cloud provider consoles.

Activities

The main activities performed by the system are:

1. **Cloud Account Onboarding**
    - Accepts read-only cloud credentials.
    - Validates connectivity to the selected cloud provider.
    - Discovers available storage resources.
2. **Configuration Retrieval**
    - Retrieves cloud storage configuration data (permissions, encryption, logging, policies) via secure APIs.
    - Holds raw configuration data in memory only for the duration of the scan.
3. **Rule-Based Misconfiguration Analysis**
    - Evaluates retrieved configurations against predefined security rules.
    - Identifies misconfigurations such as public access, weak permissions, missing encryption, and disabled logging.
4. **ML-Based Anomaly Detection**
    - Loads synthetic access logs.
    - Applies an unsupervised machine-learning model to identify anomalous access behavior.
    - Produces anomaly findings with associated scores.
5. **Severity Classification and Aggregation**
    - Assigns severity levels (Low, Medium, High) to all findings.
    - Aggregates rule-based and ML-based results into a unified findings set.

6. **Reporting and Visualization**
    - Presents findings in a human-readable format.
    - Supports filtering by severity, provider, and finding type.
    - Allows export of reports in structured formats.
7. **Scheduling**
    - Supports periodic automated scans based on user-defined schedules.

Data
The system processes and manages the following categories of data:

- **Cloud Configuration Data -** Includes storage permissions, encryption settings, access policies, and logging configuration.
  This data is treated as sensitive and is stored **in memory only** during scan execution.
- **Synthetic Access Logs -** Artificially generated log entries representing access activity (e.g., user, timestamp, action, source).
  Used exclusively for ML-based anomaly detection and evaluation.
- **Processed Findings -** Non-sensitive results produced by rule-based and ML-based analysis, including severity, description, remediation recommendation, and metadata.
  These findings may be persisted in a local database.
- **Scan Metadata -** Information about scan execution such as timestamps, trigger type (manual or scheduled), and scan status.

b. Main System Internal Scenario

Main Scenario: End-to-End Automated Security Scan
The main internal scenario of the system is the execution of a full automated security scan, which represents the core functionality of the system.

1. **Scan Trigger -** A scan is initiated either manually by a user through the interface or automatically by the scheduling mechanism.
2. **Scan Initialization -** The system creates a new scan context and initializes internal structures for storing intermediate results and metadata related to the scan execution.
3. **Configuration Retrieval -** For each onboarded cloud account, the system connects to the corresponding cloud provider using read-only credentials and retrieves storage configuration data, including permissions, encryption settings, logging configuration, and access policies.
   All retrieved configuration data is kept in memory only for the duration of the scan.

4. **Rule-Based Misconfiguration Analysis -** The retrieved configuration data is passed to the rule-based analysis component.
   Each storage resource is evaluated against a predefined set of security rules, and detected misconfigurations are converted into findings with predefined severity levels.

5. **Synthetic Log Processing -** The system loads synthetic access logs associated with the scan and preprocesses them into a suitable format for analysis.

6. **ML-Based Anomaly Detection -** The machine-learning component analyzes the processed synthetic logs using an unsupervised model to identify anomalous access patterns.
   Suspicious entries are flagged and converted into anomaly findings with associated anomaly scores.

7. **Aggregation and Severity Assignment -** Rule-based findings and ML-based anomaly findings are aggregated into a unified findings set. Each finding is assigned a final severity level (Low, Medium, High) based on rule definitions or anomaly score thresholds.

8. **Persistence of Results -** The system stores only processed findings and scan metadata in the local database.
   No raw configuration data or sensitive credentials are persisted.

9. **Report Generation -** A consolidated, human-readable report is generated summarizing all findings, grouped by severity and resource.

10. **Scan Completion -** The scan context is finalized, and the system transitions back to an idle state, ready to accept new scan requests or scheduled executions.

c. Additional System Internal Scenarios

Scenario 1: Cloud Account Onboarding

The system receives cloud account details and read-only credentials from the user interface.

1. The system validates the credential format and attempts a secure connection to the selected cloud provider.
2. Upon successful connection, the system retrieves a minimal preview of available storage resources to verify access.
3. Cloud account metadata is stored for future scans, while credentials are handled securely and not persisted.
4. The system marks the account as ready for on-demand or scheduled scans.

Scenario 2: Findings Viewing and Filtering

1. The system receives a request to view scan findings.
2. Stored findings are retrieved from the database based on user-defined filters such as severity level, cloud provider, and finding type.
3. The system formats the filtered findings into a readable representation.
4. Detailed information and remediation recommendations for selected findings are returned to the interface.

Scenario 3: Report Export

1. The system receives a request to export scan results.
2. Findings associated with the selected scan or time range are loaded from the database.
3. Findings are grouped by severity and formatted into the requested output format (e.g., JSON or text).
4. The generated report is made available for download.

Scenario 4: Scheduled Scan Execution

1. The system activates a scheduled scan based on predefined recurrence rules.
2. At the scheduled time, the system automatically initiates the main scan workflow.
3. Scan execution and result handling follow the same process as a manually triggered scan.
4. The system records execution metadata and updates scan history accordingly.

3. Design

a. **Data Design** - Database Description

Notation:
PK (Primary Key) uniquely identifies a record.
FK (Foreign Key) references a primary key in another entity.

Database Purpose - The database stores **processed scan results and operational metadata** to support viewing, filtering, historical tracking, and report export.
Sensitive data such as raw cloud configurations and credentials are **not stored**.

Storage Approach - A lightweight relational database (e.g., SQLite) is used to ensure simplicity, portability, and structured querying.

Core Data Entities

Cloud Account:

- cloud_account_id (PK)
- provider (AWS / Azure / GCP)
- account_alias
- created_at
- last_connection_status

ScanRun

- scan_run_id (PK)
- cloud_account_id (FK → CloudAccount)
- trigger_type (Manual / Scheduled)
- start_time
- end_time
- status
- summary_counts_json (optional)

StorageResource

- resource_id (PK)
- cloud_account_id (FK → CloudAccount)
- resource_name
- resource_region
- resource_type

Finding

- finding_id (PK)
- scan_run_id (FK → ScanRun)
- cloud_account_id (FK → CloudAccount)
- resource_id (FK → StorageResource, nullable)
- finding_type (Misconfiguration / Anomaly)
- severity (Low / Medium / High)
- title
- description
- remediation_recommendation
- rule_id / anomaly_score (nullable)
- metadata_json

Data Handling Constraints

- Raw configuration data and credentials are kept in memory only.
- Persisted data contains no sensitive information.
- Stored findings are used exclusively for analysis, visualization, and reporting.

b. **Data Flow** – Data Flow Between System Modules
This section describes how data flows between the main system modules during normal operation.

1. Scan Initiation Flow

- A scan request is triggered either manually through the user interface or automatically by the scheduling component.
- The request is passed to the **Scan Orchestrator**, which creates a new scan context.

2. Configuration Retrieval Flow

- The Scan Orchestrator invokes the appropriate **Cloud Connector** for each onboarded cloud account.
- Cloud Connectors retrieve storage configuration data (permissions, encryption, logging, policies) from cloud provider APIs.
- Retrieved configuration data is transferred **in memory** to the Rule Engine and is not persisted.

3. Rule-Based Analysis Flow

- The **Rule Engine** receives in-memory configuration data.
- Each storage resource is evaluated against predefined security rules.
- Detected misconfigurations are converted into structured findings and returned to the Scan Orchestrator.

4. Synthetic Log Processing Flow

- The Scan Orchestrator loads synthetic access logs through the **Synthetic Log Loader**.
- Preprocessed log data is passed to the **ML Engine** for analysis.

5. ML-Based Anomaly Detection Flow

- The ML Engine analyzes log data using an unsupervised model.
- Anomaly scores are computed and suspicious entries are converted into anomaly findings.
- Anomaly findings are returned to the Scan Orchestrator.

6. Aggregation and Persistence Flow

- The Scan Orchestrator aggregates rule-based and ML-based findings.
- Aggregated findings and scan metadata are passed to the **Findings Service**.
- Only processed findings and metadata are stored in the database.

7. Results Presentation and Export Flow

- The user interface requests findings or reports from the **Findings Service** and **Reporting Service**.
- Retrieved data is formatted for display or export (JSON or text) and returned to the user.

c. **Structural Design** - Class Diagram

The system is composed of a set of core classes organized around scan orchestration, cloud interaction, analysis, and result management. The class structure emphasizes modularity and separation of concerns by clearly distinguishing between domain entities, analysis components, and control and service logic.

Core Domain Classes

The core domain classes represent the primary data entities managed by the system and encapsulate state and relationships without embedding business logic.

- **CloudAccount -** represents an onboarded cloud account and its associated metadata, including provider information and connection status. A single CloudAccount may be associated with multiple storage resources and multiple scan executions.
- **StorageResource** - represents a cloud storage resource, such as a bucket or container, whose configuration is analyzed during scans. Each StorageResource belongs to exactly one CloudAccount.
- **ScanRun** - represents a single execution of a security scan and tracks its lifecycle, including timing and execution status. Each ScanRun is associated with one CloudAccount and may produce multiple findings.
- **Finding** - represents a detected security issue, either a configuration misconfiguration or an anomalous access pattern. A Finding is linked to a specific ScanRun and may optionally be associated with a particular StorageResource.

Analysis and Processing Classes

Analysis and processing classes encapsulate the system's detection logic and are responsible for identifying security risks.

- **Rule** - represents a single misconfiguration detection rule and defines the logic required to evaluate a storage resource configuration. Each rule may produce zero or one finding when evaluated.
- **RuleEngine** - applies a collection of Rule objects to retrieved storage resource configurations and aggregates the resulting findings.
- **MLModel** - represents an unsupervised machine-learning model used to analyze synthetic access logs and identify anomalous access behavior, producing anomaly-related findings.

Control and Service Classes

Control and service classes coordinate system behavior and manage interactions between components.

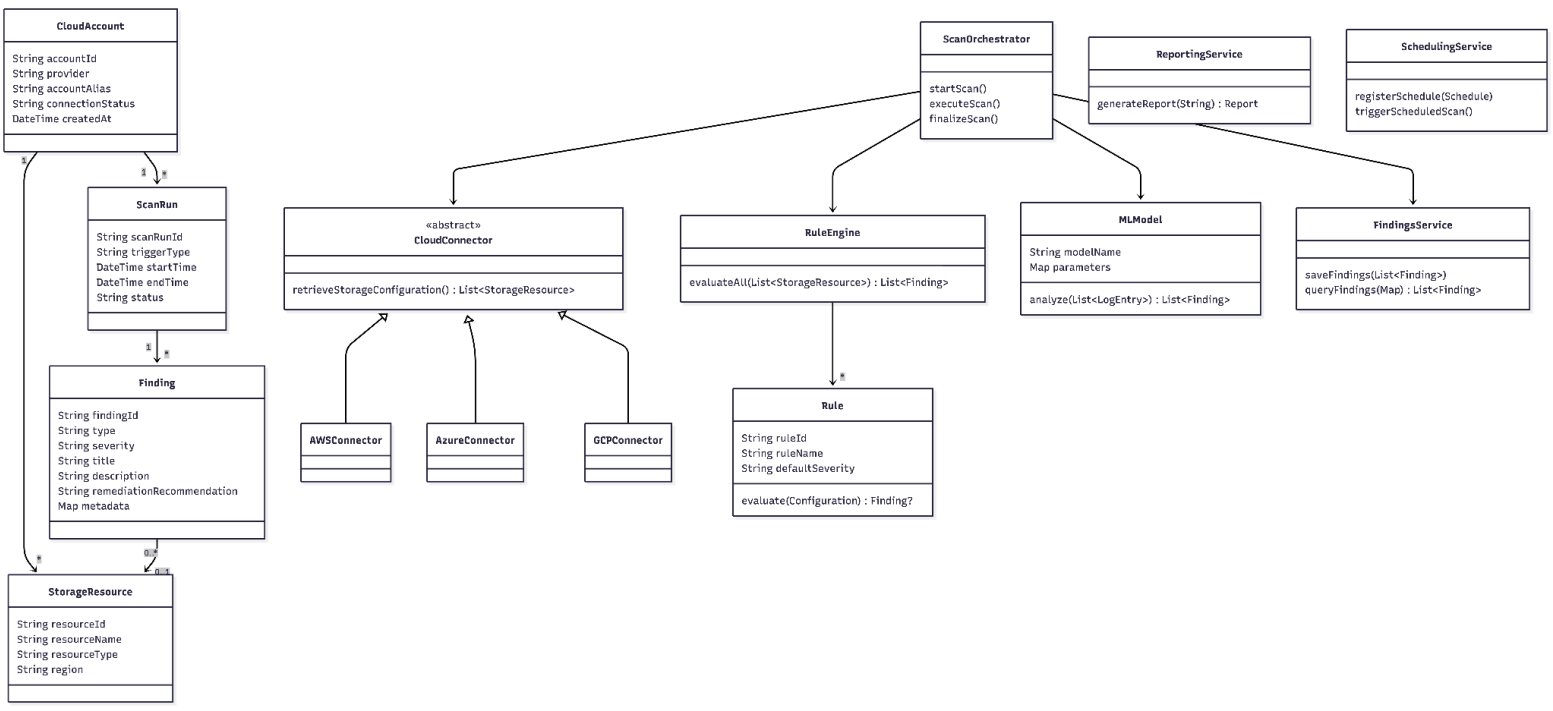
- **ScanOrchestrator** - coordinates the end-to-end scan execution process, including configuration retrieval, rule-based analysis, anomaly detection, and result aggregation. It relies on cloud connectors, analysis components, and persistence services.
- **CloudConnector** - is an abstract component responsible for handling communication with cloud provider APIs. Provider-specific implementations (AWSConnector, AzureConnector, GCPConnector) inherit from this abstraction to isolate cloud-specific logic.
- **FindingsService** - manages persistence and retrieval of processed scan findings.
- **ReportingService** - generates human-readable and structured reports from stored findings.
- **SchedulingService** - manages the configuration and execution of scheduled scans.

Design Characteristics

The class structure enforces a clear separation between domain entities, analysis logic, and control services. Cloud-provider-specific functionality is isolated through abstraction, and analysis components are designed to be replaceable and extensible. The structure maps directly to a UML class diagram without ambiguity.

UML Representation

The following UML class diagram illustrates the structural components of the system and their relationships as described above.



d. Interactions Design

This section describes how system components interact during execution, based on the use cases and system flows defined in the SRS.
The interactions are presented at a design level, focusing on component collaboration rather than user interface details.

i. Use Cases

The system supports the following use cases, as defined in the SRS, and these form the basis for all runtime interactions.

UC1 – Cloud Account Onboarding

Actor: Security Admin
The user provides read-only cloud credentials and selects a cloud provider.
The system validates the credentials by attempting a secure connection through the appropriate CloudConnector.
Upon successful validation, the cloud account metadata is registered, and the system becomes ready to perform on-demand or scheduled scans.

UC2 – Automated Security Scan
Actor: Security Admin (manual) / System (scheduled)
A scan is triggered either manually by a user or automatically by the scheduling mechanism.
The system retrieves cloud storage configurations, performs rule-based misconfiguration analysis, optionally analyzes synthetic access logs using the ML component, aggregates findings, and stores the processed results.

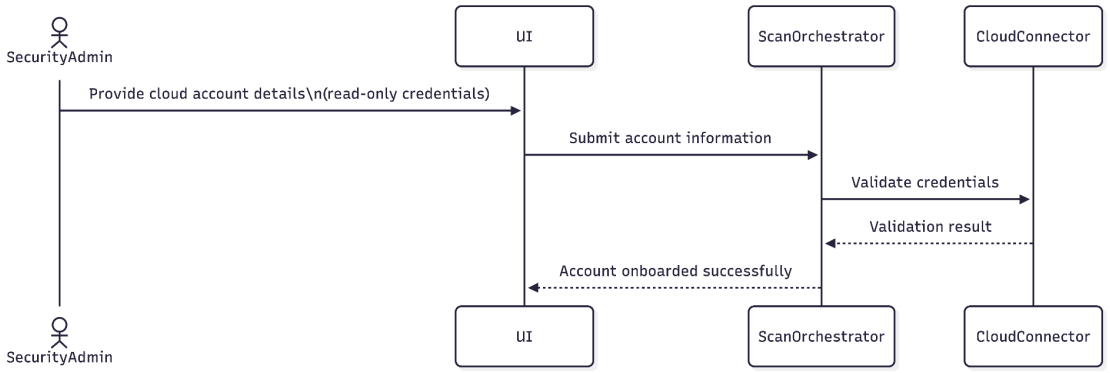
UC3 – Investigate Security Finding
Actor: Security Analyst
The user requests to view scan findings and applies filters such as severity, provider, or finding type.
The system retrieves the relevant findings from persistent storage and presents detailed descriptions, severity classification, and remediation recommendations.

UC4 – Export Results Report
Actor: Compliance Officer
The user requests an export of scan results for a selected period or scan run.
The system groups stored findings by severity and generates a report in the requested format (e.g., JSON or text), which is then made available for download.

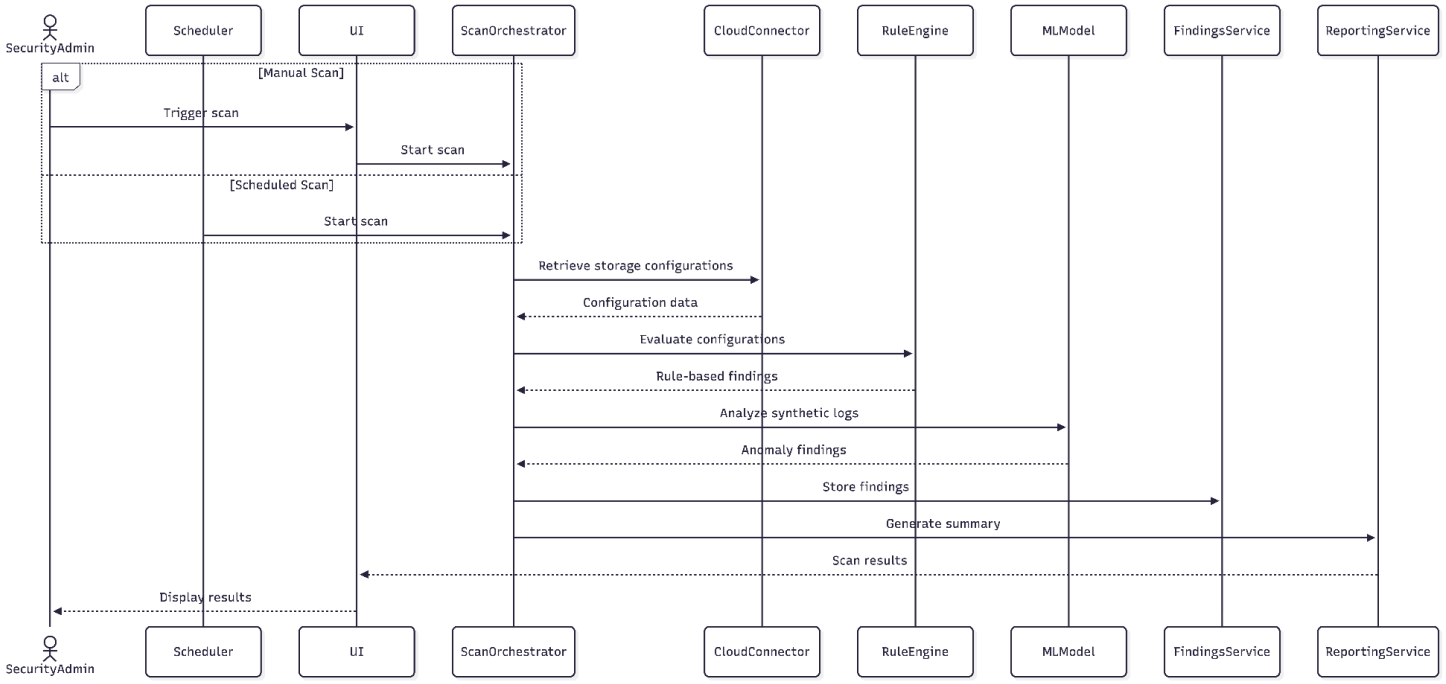
UC5 – Configure Scan Scheduling
Actor: Security Admin
The user defines a recurring scan schedule (e.g., daily or weekly).
The system validates the schedule configuration and registers it with the scheduling component, enabling automatic scan execution.

ii. Sequence Diagram

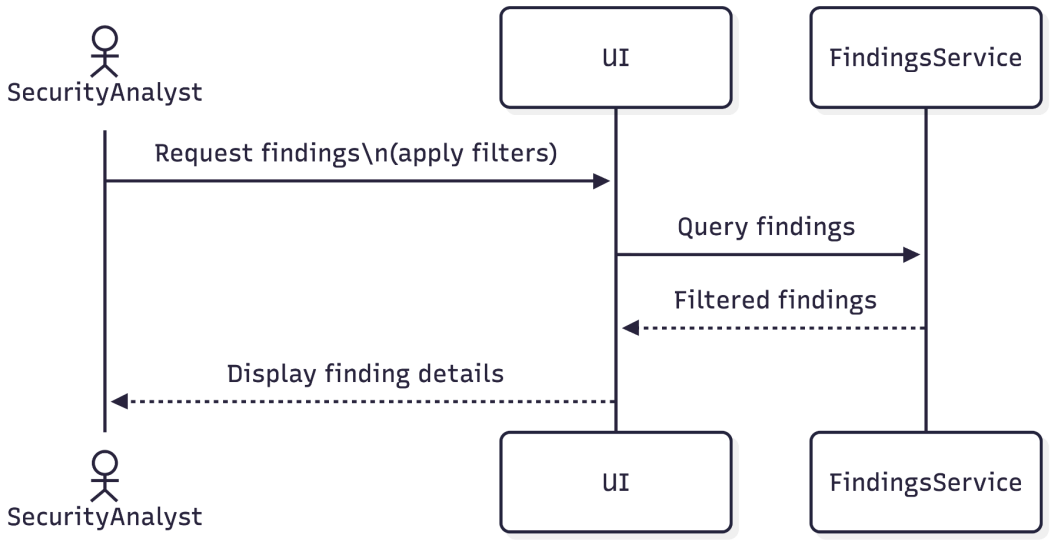
UC1 – Cloud Account Onboarding



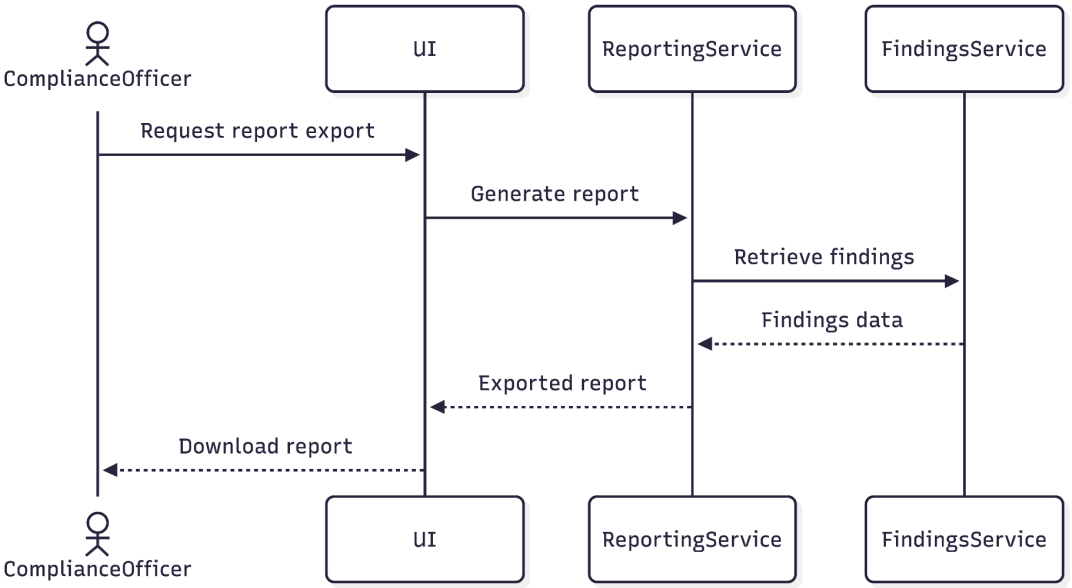
UC2 – Automated Security Scan



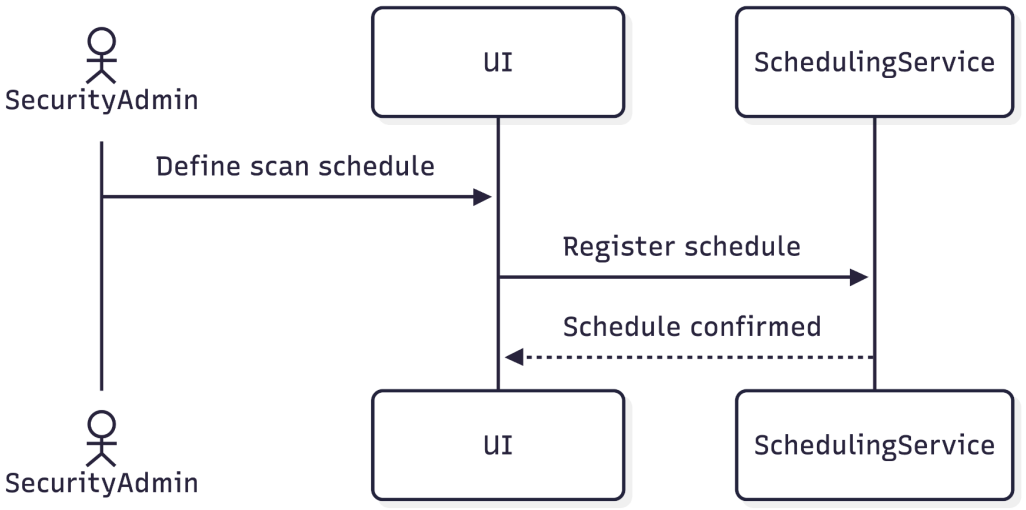
UC3 – Investigate Security Finding



UC4 – Export Results Report



UC5 – Configure Scan Scheduling



iii. Activity Diagram / State / Processes

This subsection describes the main operational processes of the system using activity diagrams.

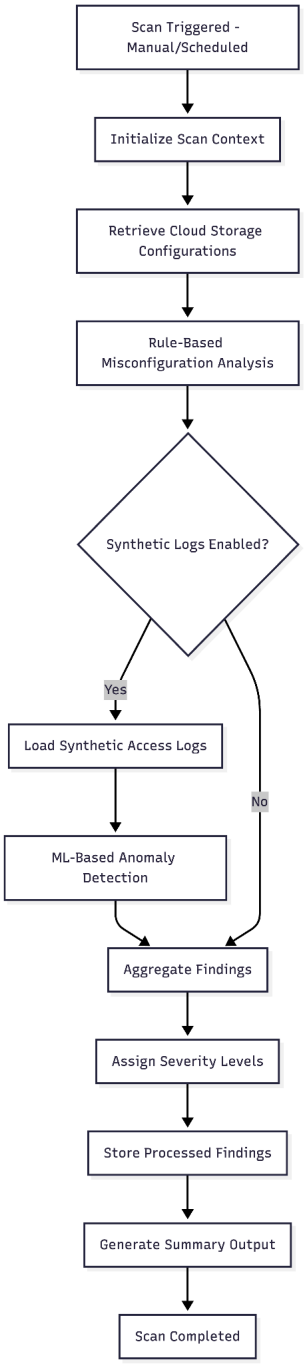
The activity flows are derived directly from the system flows defined in the SRS and illustrate the sequence of actions performed by the system during execution, without focusing on user interface details or implementation-specific logic.

The primary process is the **automated security scan**, supported by secondary processes for findings review, report export, and scan scheduling configuration

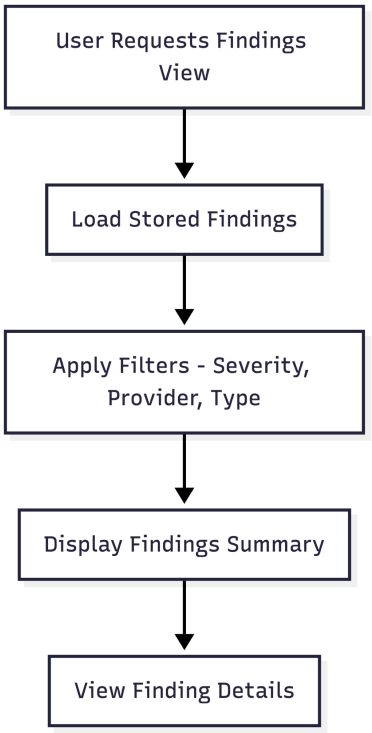
Automated Security Scan – Activity Diagram

This activity diagram represents the end-to-end scan execution process and corresponds to **UC2 and System Flow 1** in the **SRS**.

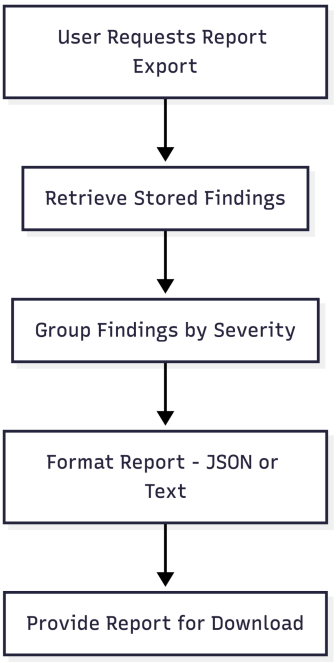
This process highlights the sequential stages of scan execution and the optional execution of machine-learning-based anomaly detection.



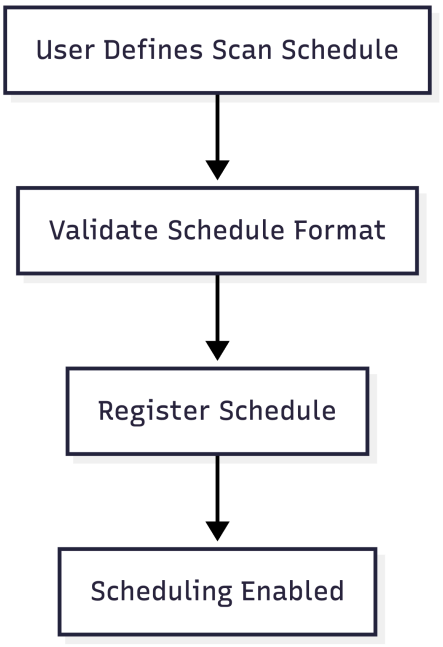
Findings Viewing and Filtering – Activity Diagram
This activity diagram corresponds to **UC3 and System Flow 2** in the **SRS**.
This process illustrates how stored findings are retrieved and presented without interacting with cloud provider APIs.



Report Export – Activity Diagram
This activity diagram corresponds to **UC4 and System Flow 3** in the **SRS**.
The diagram emphasizes the separation between data retrieval, report formatting, and delivery.



Scan Scheduling Configuration – Activity Diagram
This activity diagram corresponds to **UC5** in the **SRS**.
This process focuses solely on scheduling configuration, while actual scan execution is handled by the automated scan process.



The activity diagrams collectively describe the system's operational behavior, covering both the primary scanning workflow and supporting processes.
They provide a clear view of control flow, decision points, and processing stages, while remaining consistent with the SRS-defined system flows.

e. **Description of Algorithmic Components**

This section describes the main algorithmic components of the system that are responsible for detecting security misconfigurations and suspicious activity.
The system combines rule-based analysis with unsupervised machine-learning-based anomaly detection to provide comprehensive coverage of cloud storage security risks.

Rule-Based Misconfiguration Detection
The rule-based detection component evaluates cloud storage configurations against a predefined set of security best practices.
Each rule encapsulates a single misconfiguration condition and produces a finding when a violation is detected.
Typical rule categories include:

- Public access exposure (e.g., publicly readable or writable storage)
- Missing or incorrectly configured encryption
- Excessive or overly permissive access policies
- Disabled or insufficient logging configuration

Rules are applied independently to each storage resource, allowing findings to be generated in a deterministic and explainable manner.
Each detected issue is assigned a default severity level (Low, Medium, High) based on its potential impact and exposure.

(This component directly implements **FR4 - FR8** from the SRS.)

Unsupervised ML-Based Anomaly Detection
The anomaly detection component analyzes synthetic access logs to identify unusual or suspicious access behavior that may indicate misuse or potential data leakage.

The system employs a lightweight unsupervised learning approach, suitable for scenarios where labeled data is unavailable.
The model is trained or configured on synthetic baseline behavior and computes anomaly scores for individual log entries or aggregated sessions.

Key characteristics of the ML component:

- Operates only on synthetic datasets
- Does not require labeled training data
- Produces anomaly scores rather than binary decisions
- Flags events exceeding predefined anomaly thresholds

Detected anomalies are converted into findings and integrated with rule-based results, enabling unified reporting and severity classification.

(This component directly implements **FR9 - FR11** from the SRS.)

Severity Classification and Aggregation

Both rule-based and ML-based findings are passed through a common aggregation and classification process.

Findings are normalized into a unified structure and assigned final severity levels based on:

- predefined rule severity (for misconfigurations), or
- anomaly score thresholds (for ML findings).

This unified handling ensures consistent prioritization and reporting across different detection mechanisms.

Design Considerations

The algorithmic components are designed to be:

- **Modular** – rules and ML models can be added, removed, or replaced independently
- **Explainable** – rule-based findings include explicit causes and remediation guidance
- **Lightweight**
- **Extensible** – capable of supporting additional detection logic or models in the future

The combination of deterministic rule-based analysis and unsupervised anomaly detection allows the system to detect both known misconfiguration risks and unexpected access patterns.

This hybrid approach balances explainability, flexibility, and practical feasibility within the defined project scope.

f. Software Architecture Pattern

i. N-tier: Data, Logic, Service, Presentation tiers etc.

The system is designed according to a **layered (N-tier) software architecture**, which separates responsibilities across distinct logical tiers.

This architectural pattern supports modularity, maintainability, and extensibility, and aligns with both the functional and non-functional requirements defined in the SRS.

N-Tier Architecture

The system is organized into the following logical tiers:

- Presentation Tier - provides the user-facing interface for interacting with the system.
  It enables users to trigger scans, view and filter findings, configure scan scheduling, and export reports through a simple CLI or minimal web interface.
  This tier contains no business logic and communicates exclusively with the service tier.
- Service / Control Tier - The service tier coordinates system operations and manages interactions between architectural components.
  Its responsibilities include:
  - ❖ Orchestrating scan execution.
  - ❖ Managing scheduled scans.
  - ❖ Handling persistence and retrieval of processed findings.
  - ❖ Generating human-readable and structured reports.

  Core components in this tier include the **ScanOrchestrator**, **SchedulingService**, **FindingsService**, and **ReportingService**.
- Logic / Analysis Tier - The logic tier contains the core detection and analysis mechanisms of the system.
  It includes:
  - ❖ The **RuleEngine** and individual **Rule** components for rule-based misconfiguration detection.
  - ❖ The **MLModel** component for unsupervised anomaly detection based on synthetic access logs.

  This tier is independent of presentation and persistence concerns, allowing detection logic to evolve without impacting other system layers.
- Data Tier - The data tier is responsible for storing processed scan findings and execution metadata.
  A lightweight relational database is used, and sensitive data such as raw cloud configurations and credentials is never persisted.
  This tier supports querying, filtering, and exporting historical scan results.

ii. Optional: MVC- Model, View, Controller structure (or else)

The selected architecture exhibits the following characteristics:

- **Separation of concerns -** each tier has a clearly defined responsibility.
- **Loose coupling -** components interact through well-defined interfaces.
- **Extensibility -** new cloud providers, detection rules, or analysis models can be integrated with minimal architectural changes.
- **Testability -** individual tiers and components can be tested independently.
- **Security-aware design -** sensitive data is processed in memory only and excluded from persistent storage.

MVC Consideration

A strict Model–View–Controller (MVC) pattern is not applied, as the system primarily focuses on backend analysis rather than complex user interaction.

However, the layered architecture conceptually aligns with MVC principles, where:

- Domain entities correspond to the **Model**,
- The presentation tier acts as the **View**,
- Service and orchestration components fulfill the **Controller** role.

The layered software architecture provides a clear and robust structural foundation for the system, supporting security analysis, scalability, and future extensibility while remaining appropriate for the defined project scope.

4. Validation

a. Validation and Evaluation Plan

This section outlines the planned approach for validating and evaluating the system's correctness, effectiveness, and robustness.
The validation strategy focuses on verifying that the system meets its functional requirements and performs reliably under controlled and repeatable conditions.
Validation is performed using **synthetic cloud configurations and access logs**, allowing controlled injection of misconfigurations and anomalous behavior.
This approach ensures repeatability, safety, and alignment with the project scope.

Rule-Based Detection Validation

To validate rule-based misconfiguration detection:

- Synthetic cloud storage configurations are created with **known injected misconfigurations**, such as public access enabled, missing or incorrect encryption, overly permissive access policies and disabled logging.
- Each injected misconfiguration is tracked as a ground-truth condition.
- Validation checks whether the system correctly detects and reports these issues.

Evaluation criteria:

- Detection coverage of injected misconfigurations
- Correct severity classification (Low / Medium / High).
- Absence of false positives on correctly configured resources.

ML-Based Anomaly Detection Validation

To validate the anomaly detection component:

- Synthetic access logs are generated to represent normal access behavior.
- Controlled anomalous patterns are injected, such as unusual access times, unexpected access sources and abnormal access frequency or volume.
- The ML model is evaluated on its ability to assign higher anomaly scores to injected anomalies.

Evaluation criteria:

- Recall of injected anomalies.
- False-positive rate on normal synthetic behavior.
- Stability of anomaly scoring across repeated runs.

System-Level Evaluation

At the system level, validation ensures that:

- Rule-based and ML-based findings are correctly aggregated.
- Findings are consistently stored and retrievable.
- Reports accurately reflect detected issues and severity levels.
- Manual and scheduled scans produce equivalent results.

The main system KPI defined in the SRS is used as a guiding benchmark:

- At least **90% recall** on injected misconfigurations
- Less than **10% false-positive rate** on the synthetic benchmark dataset

b. Testing Platform

The testing platform is designed to support reproducible and platform-independent validation.

Environment

- Operating Systems: Linux, macOS, Windows
- Execution Mode: Local execution in a controlled environment
- Cloud Interaction: Limited to test or sandbox cloud accounts, where applicable

Test Data

- Synthetic cloud storage configurations
- Synthetic access logs for anomaly detection
- Predefined rule sets for misconfiguration detection

Testing Approach

- Unit tests for individual rules and analysis components
- Integration tests covering end-to-end scan execution
- Manual validation of reports and exported outputs

The use of synthetic data ensures that testing does not expose real cloud resources or sensitive information.

5. Project Management

a. Schedule / Gantt (possible print screen or sharable link)

Project Phases

1. Requirements Analysis (10.11.25 - 19.11.25)
    - Review problem domain
    - Define scope and goals
    - Produce One-Pager and SRS
2. System Design (20.11.25 - 15.12.25)
    - Define system architecture (20.11.25 - 27.11.25)
    - Design data model and data flow (28.11.25 - 04.12.25)
    - Create UML class, sequence, and activity diagrams (28.11.25 - 04.12.25)
    - Produce the Software Design Document (SDD) (05.12.25 - 15.12.25)
3. Implementation (16.12.25 - 14.01.26)
    - Cloud API integration (16.12.25 - 25.12.25)
    - Rule-based misconfiguration detection (20.12.25 - 01.01.26)
    - Synthetic log generation (26.12.25 - 04.01.26)
    - ML-based anomaly detection (28.12.25 - 10.01.26)
    - CLI / minimal UI implementation (03.01.26 - 14.01.26)

4. Testing and Validation (10.01.26 - 27.01.26)
    - Unit testing of rules and analysis components (10.01.26 - 17.01.26)
    - Integration testing of scan execution (15.01.26 - 23.01.26)
    - Validation using synthetic datasets (19.01.26 - 27.01.26)

5. Documentation and Final Delivery (26.01.26 - 08.02.26)
    - Finalize documentation (26.01.26 - 02.02.26)
    - Prepare submission materials (30.01.26 - 02.02.26)
    - Final review and cleanup (03.02.26 - 08.02.26)

Link to the Gantt

b. Team Roles - final

The project is developed collaboratively, with responsibilities distributed as follows:

- **Jamal Tannous**
    - System architecture and design
    - Rule-based detection design and implementation
    - Integration of system components
    - Technical documentation

- **Asya Pesran**
    - Requirements analysis and validation planning
    - Synthetic data generation
    - Testing and evaluation support
    - Documentation support

- **Dan Ydov**
    - ML-based anomaly detection design
    - Model evaluation and tuning
    - Data analysis support
    - Documentation support

Team members collaborate across phases to ensure design consistency and integration quality.