**Security requirements for COMESA features development**

The following additional minimum mandatory controls as outlined in the Tazama Security Design Document (BRS) must be delivered in the COMESA features development project. The existing functionality does not require any changes and should be preserved.

The additional improvement recommendations will form part of a future roadmap and are not required to be addressed by the features development project.

**3. IAM (Identity and Access Management)**

**3.1 API Authentication**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Token-Based Authentication for All APIs:    1. Tokens are initially issued by the IAM provider (Keycloak). Tazama repackages the Keycloak JWT as a more generic Tazama JWT token.    2. Every Tazama endpoint (TMS, Admin Service) requires a valid JWT (tazamaToken).    3. To obtain this token, clients must authenticate with Keycloak (via the Auth‐service). | Existing – to be preserved |
| 1. Integration with Keycloak (OIDC): The Auth‐service acts as a confidential client to Keycloak, exchanging user credentials for a Keycloak token, then transforming that token into Tazama’s own JWT format for consistency across Tazama microservices. | Existing – to be preserved |
| 1. Cryptographic Signing of Tokens: Tazama issues tokens signed by an internal private key; each Tazama service verifies the signature with the corresponding public key. Any alteration to the token is thus detected and rejected. | Existing – to be preserved |
| 1. Central Auth‐lib for Validation: All Tazama services rely on the same Auth‐lib function (validateTokenAndClaims()) to parse, verify, and check claims, ensuring uniform access control rules. | Existing – to be preserved |
| 1. Central Auth‐lib for Validation: All Tazama services rely on the same Auth‐lib function (validateTokenAndClaims()) to parse, verify, and check claims, ensuring uniform access control rules. | Existing – to be preserved |
| 1. Tokens Include Embedded Claims: All tokens must include embedded claims defining user roles and scopes that control access to various services and functions. | Existing – to be preserved |

**3.2 User Management**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Centralized User Directory: All user accounts (including administrators) are stored in Keycloak. Tazama does not keep passwords internally, delegating to Keycloak for credential storage and verification. | Existing – to be preserved |
| 1. Roles Assigned in Keycloak: Roles (e.g., TMS\_Admin, TMS\_User) exist in Keycloak; these are included as claims in the JWT. Tazama’s services interpret these roles to permit or deny actions. | Existing – to be preserved |
| 1. Management via Keycloak Console or API: Tazama operators use Keycloak’s admin UI or admin APIs to create, update, or remove users, set passwords, and assign roles. | New – to be delivered |
| 1. Separation of Administrative Duties: The TMS and Admin Service are restricted to the necessary roles. Complete platform or realm‐wide user tasks (like creating new admin roles) remain inside Keycloak’s console. | Existing – to be preserved |
| 1. Basic Lifecycle Processes: Administrators can revoke user credentials, reset passwords, or deactivate accounts via Keycloak. Tazama honors the updated claims once the user obtains a new token. | Existing – to be preserved |

**3.3 Service‐to‐Service Authentication**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Auth‐Service to Keycloak: The Auth-Service uses a secure client secret (stored outside the codebase) to authenticate with Keycloak’s token endpoint. Keycloak only issues tokens to recognized clients. | Existing – to be preserved |
| 1. User Token Propagation: If one Tazama microservice calls another “on behalf of a user,” it forwards the same user JWT so the callee can re‐check roles. This preserves the end‐user context throughout the microservice chain. | New – to be delivered |
| 1. Private Network Boundaries: Internal Tazama components (Event Director, Rule Processors, etc.) communicate on a private cluster network. They typically do not expose endpoints to the public internet. | New – to be delivered |
| 1. All Internal Service Calls Use JWT: All service-to-service communication within Tazama must use JWT-based authentication | Existing – to be preserved |

**4. Data Security**

**4.1 Role‐Based Access Control (RBAC)**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Role‐to‐Privilege Mapping: Each Tazama endpoint or function checks for specific roles. For example, only users with an “Admin” role can alter rules or typologies. | Existing – to be preserved |
| 1. Enforcement in Microservices: The Auth‐lib automatically denies requests unless the JWT includes the required role claim. This check runs on every request to TMS, Admin, or another service. | Existing – to be preserved and new |
| 1. Default Deny Posture: New users have no roles, ensuring they cannot inadvertently access Tazama features without explicit assignment. | Existing – to be preserved |
| 1. Admin/User Role Segregation: Administrative tasks (like modifying system config) are not included in standard user roles, preventing accidental overreach. | Existing – to be preserved |
| 1. Documented Roles: The Tazama repository or documentation includes definitions of each role (e.g., TMS\_User, TMS\_Admin) and the associated access rights. | New – to be delivered |
| 1. Granular Role Expansion: As Tazama adds more modules or new features (e.g., a case management UI), ensure each major capability corresponds to a distinct role. | New – to be delivered |
| 1. Segregation of Duties Enhancements: For high‐risk changes (like rules that can block transactions), separate those privileges so no single user can create and override without a second approval. | New – to be delivered |
| 1. UI Role Awareness: If Tazama or external front‐ends are used, hide or turn off UI elements unless the user’s token has the relevant role. This reduces error rates and user confusion. | New – to be delivered |

**4.2 Least Privilege**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Scoped Service Responsibilities: The TMS API handles transaction ingestion and data prep, the Rule Processor only runs its logic, etc. Each microservice has a narrowly defined role. | Existing – to be preserved |
| 1. **Separate Credentials for Each Service**: The TMS API, Admin Service, and other components do not share DB credentials. If one is compromised, it does not grant universal data access. | New – to be delivered |
| 1. **Admin Function Isolation**: Configuration tasks (like turning a typology on or off) reside in the Admin Service, not the TMS. This inherently prevents user‐level endpoints from making admin changes. | New – to be delivered |
| 1. Minimal Data Retrieval: Rule processors pull only the historical data needed for their specific check, typically via DataCache or targeted DB queries. They do not fetch entire account sets. | Existing – to be preserved |
| 1. Lock Down Container Privileges: Ensure Tazama containers run as non‐root, with read‐only file systems where possible, and no extra Linux capabilities. | New – to be delivered |

**4.3 Strong Database Authentication**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Unique Database Accounts: Each service has a DB username/password. The application code does not use a “root” or “superuser” DB account, limiting damage from stolen credentials. | New – to be delivered |
| 1. Secure Storage of Credentials: DB passwords and keys are never in source control. They are set at deployment time (e.g., via Kubernetes Secrets). | Existing – to be preserved |
| 1. Configured Password Complexity: Long, random passwords are used for DB accounts. If a breach occurs, credentials can be rotated by updating secrets and redeploying. | Existing – to be preserved |
| 1. Authentication Enforced by ArangoDB: The DB does not accept unauthenticated connections. Each service must provide valid credentials. | Existing – to be preserved |
| 1. Logging and Alerting on Failed DB Logins: Feed ArangoDB’s auth logs into the central monitoring system and create alerts if repeated failures occur. | New – to be delivered |
| 1. Granular DB Roles: If a service only needs read access to specific collections, do not grant write or admin privileges. This enforces least privilege at the DB level. | New – to be delivered |
| 1. Network‐Level DB Restrictions: Only allow connections to ArangoDB from Tazama’s Kubernetes namespace or VPC. Deny external traffic to the DB port entirely. | Existing – to be preserved |

**4.4 Data at Rest**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Database Encryption: ArangoDB’s data files and backups are encrypted (e.g., using disk encryption on the underlying host or ephemeral volumes in a cloud environment). | New – to be delivered |
| 1. Encrypted Secrets in Cluster: Kubernetes’ etcd (where secrets might live) is configured with encryption enabled, preventing local reading of secrets from disk in plaintext. | Existing – to be preserved |
| 1. Encrypted Backups: Tazama backups (database dumps, config exports) are stored encrypted, typically in a restricted S3 bucket or similarly secured location. | New – to be delivered |
| 1. Minimal File System Permissions: Each microservice can only read/write the required directories. World‐readable or world‐writable files are avoided. | Existing – to be preserved |
| 1. Regular Encryption Audits: Periodically verify that new data sets or extended features follow encryption policies. Confirm that backups are restorable and remain encrypted in transit and storage. | New – to be delivered |

**4.5 Data in Transit**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Auth‐service <-> Keycloak Over HTTPS: User credentials or tokens are never sent in plaintext. Keycloak’s certificate is validated to prevent spoofing. | New – to be delivered |
| 1. Internal Pod Security Assumptions: Within the Kubernetes cluster, Tazama often uses private networking for microservice communication (Event Director to Rule Processor, etc.). | Existing – to be preserved |
| 1. Avoid Plaintext Sensitive Data: Even in internal communications or logs, Tazama ensures that especially sensitive values (e.g., user passwords, entire tokens, PII) are not transmitted in the clear or logged in the open. | New – to be delivered |

**4.6 SQL Injection Protection**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Parameterized Queries: To prevent SQL injection attacks, all database queries must use parameterized statements. As a guideline:    * Arango Query Language (AQL) provides for AQL-safe parameterization in TypeScript by using a template string generator function, e.g. <https://github.com/frmscoe/rule-054/blob/434a5fe040691dad84e7c8352edfa3e6a5121e17/src/rule-054.ts#L59> | Existing – to be preserved |
| 1. ORM Usage: Use ORM frameworks that automatically implement safe query practices. | New – to be delivered |

**4.7 Input Validation**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. API Boundary Validation: All user input must be validated at the API boundary before processing. | Existing – to be preserved |
| 1. Type Checking: Implement strong type checking for all inputs to prevent type confusion attacks. | New – to be delivered |
| 1. OpenAPI Schemas: Use OpenAPI schemas for the structured validation of all API inputs. | Existing – to be preserved |

**4.8 Separate Environments**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Environment Separation: Development, test, and production environments must be logically and physically separated. | New – to be delivered |
| 1. Controlled Data Flow: Control data flow between environments to prevent leakage of production data. | New – to be delivered |

**5. Platform Logging, Auditing, and Security Monitoring**

**5.1 Activity Logging**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Comprehensive Request Logging: The TMS API, Admin Service, and Auth‐service record each request with timestamps, endpoint details, user identity (when present), and success/failure results. | New – to be delivered |
| 1. Centralized Aggregation: Logs are typically collected in Elasticsearch or a similar system, preventing data loss from container restarts. Operators can query or visualize all logs in one place. | Existing – to be preserved |
| 1. Structured and Correlated Logs: Tazama uses correlation or trace IDs to link events across microservices (e.g., TMS -> Event Director -> Rule Processor). This makes debugging or forensic analysis more efficient. | Existing – to be preserved |
| 1. Error/Exception Logging: JWT validation failures, DB connection errors, or rule processing exceptions are recorded with enough detail to diagnose problems (though without disclosing private keys or passwords). | New – to be delivered |
| 1. Avoid Logging Sensitive Information: Tazama does not store full tokens, user passwords, or personal data in logs. Instead, partial identifiers or masked data are used where possible. | New – to be delivered |

**5.2 Audit Trails**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Admin Action Logging: When administrators create new user accounts or modify roles (in the Admin Service or Keycloak), the event is logged. This ensures an auditable record of who changed what. | New – to be delivered |
| 1. **ArangoDB Write Accountability**: Tazama logs DB writes in each service’s operational logs, so one can trace which service performed which write operation. | Not required at this time |
| 1. **Immutable Log Storage**: Once logs or audit records reach Elasticsearch or another aggregator, they cannot be altered or deleted without leaving a trace, preserving data integrity. | New – to be delivered |
| 1. Dedicated Audit‐Log Mechanism: Consider implementing an append‐only audit database or ledger for critical events, ensuring tamper‐evident properties. | New – to be delivered |

**5.3 Integrated Application Security Monitoring**

**Minimum Mandatory Controls**

|  |  |
| --- | --- |
| Description | Requirement |
| 1. Logging Sidecar & Dashboards: Tazama uses a sidecar pattern or direct log shipping to collect logs into Elasticsearch or a similar aggregator, giving real‐time operational visibility. | Existing – to be preserved |
| 1. Rule & Typology Event Logging: Each rule or typology evaluation logs relevant details, enabling the correlation of suspicious patterns or repeated triggers across transactions. | New – to be delivered |
| 1. Basic Performance Monitoring: CPU usage, memory, and request throughput are tracked to spot potential denial‐of‐service attempts or resource exhaustion. | New – to be delivered |
| 1. Continuous Tuning: Update detection rules and thresholds as Tazama’s environment changes, ensuring minimal false positives while still catching genuine threats. | New – to be delivered |