Mojaloop data at rest (Databases)

Mojaloop data at rest include information from the following sources:

1. Application logs
2. System and infrastructure configuration files
3. System and Application logs
4. Databases

The standard's foundation is a set of important database security events and actions derived from and mapped to compliance and security requirements critical for payment providers. These compliance standards include:

1. PCI DSS
2. SOX/COBIT
3. ISO 27001
4. NIST 800-53
5. CIS Benchmarks

Some common database vulnerabilities as seen from a STRIDE threat model include:

	STRIDE (See Appendix)					
VULNERABILITY	**S**	**T**	**R**	**I**	**D**	**E**
Access control (no least privilege)	X	X	X	X		X
SQL injection		X		X		
Weak audit trails	X	X	X	X		
Denial of service					X	
Communication protocol vulnerabilities (encryption)				X		
Weak authentication	X					X
Inadequate recovery options					X	

Mojaloop environment databases:

The following are the application datastores identified in Mojaloop and their associated data:

1. Redis - Cache data
2. MongoDB - Central Event Processor Data

3. MySQL - Transactional Data
    a. Central Ledger
    b. Account Lookup
    c. Quoting Service
    d. Central Settlements
    e. Transaction Requests
4. Central Event Processor
5. Kafka - Stream information
6. Docker / Kubernetes volumes
7. EFK for security and application event logging
8. Prometheus / Grafana for application uptime monitoring
9. WSO2 - MySQL Database storing user and API authentication information and secrets management.
10. Pathfinder used by Account lookup service.

This document assumes community versions of databases; hence recommendations may include some features available only in enterprise versions of the same databases (MySQL, Redis, Percona).

Recommended standards for Databases

Data at rest is static data stored on hard drives archived or not accessed or modified. The following protective controls are recommended for Mojaloop data at rest:

1. Conduct a data classification exercise to identify types of data at rest within Mojaloop, where they are stored and access controls for each data type.

1. Access control restrictions - Enforce access control restrictions using any of the following:
    a. Administrator console for databases. This includes Redis
    b. Configure role-based access for database users, especially application-wide users that are tied to running services.
    c. Disable Redis database API unless explicitly needed to run services for cloud deployments
    d. When you enable the Redis database API, ensure the API keys for all your Redis implementation owners are securely stored and managed. These can be used to configure and administer the database programmatically.
    e. Implement MFA (multi-factor authentication) for database administrator console access.
    f. Configure IP based restrictions on which source IPs can communicate directly with the databases to post/query data
    g. Enable TLS encryption on connections for all services transmitting sensitive data to/from the database. This may impact performance but useful in implementations where PII data needs to be protected, e.g. GDPR.
    h. Create entitlement reports that can be used to review and grant/revoke access to the database. If the entitlement needs management approval through a ticketing system,

adding, suspending, deleting or modifying entry should show up in the entitlement report. The compliance department can validate the requests and approval for each entitlement added, revoked or modified.

2. Exploring PII data encryption to render it unreadable and unusable outside Mojaloop environment should a database export/dump be executed. Options here include data encryption at rest and full database encryption.
3. Configure replication and auto-failover recovery options in the event of database failure. These should be supported by appropriate backups to maintain point in time data.
4. Enable audit logging for all CRUD actions on sensitive data and where applicable ship logs to a SIEM for monitoring.
5. Harness supporting databases' publisher/subscriber (Pub/Sub) or Message Broker ability can help build the foundation of microservices architecture. It helps in decoupling the services and makes their communication more effective. Redis supports this design.
6. Run the database with least privileges. This decreases the potential damage that an attacker can cause if the session is hijacked, and the attacker gains access to system-wide administrative rights.
7. Review database code submitted by developers for any malicious code. Ensure developers are familiar with SQL injection prevention guidelines such as these: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
8. Perform database hardening activities:
    a. Remove default credentials and authentication artefacts, e.g. test database.
    b. Configure appropriate file system permissions for critical database files
    c. Ensure passwords are not stored in global configuration files
    d. Ensure local authentication bypass is not enabled for any database instance
    e. Disable server-side scripting
    f. Install relevant patches for the databases as per change management procedures

Recommended Auditable events and audit record content standard

For audit logs to be useful, they must record sufficient information to serve the operational needs, preserve accountability, and detect malicious activity. This standard defines these events and content recommended to be captured in a Mojaloop implementation.

Database systems should produce audit logs for the following key event types, depending on system capabilities:

2. Login and logoff events, including unsuccessful logons.

3. Modification of authentication mechanisms and security policies on a database, logging, configuration settings, especially activities that reduce security levels and access controls.

4. User management activities such as create user, modify user, delete user, create the role, modify a role, delete a role, assign a role, grant/revoke roles or privileges

5. Queries related to users and associated authentication policy.

6. Privileged commands
7. Object ownership changes
8. Access to sensitive data (this needs the organization to have already classified its data), including failed access.
9. Changes to the database schema (DDL (Data Definition Language) commands)
10. Database backup and restore operations
11. Attempts to access Operating system commands and functions from the database (execute commands, read/modify files and settings)

Monitoring use cases

From the audit logs captured, below are some monitoring use cases that could be configured to ensure visibility into security events within databases:

1. User account additions and changes should be reconciled against an account
2. request and approval log
3. Significant instances of failed password attempts and against multiple accounts
4. within a short time frame which may indicate hacking attempts
5. Notable examples of failed access attempts to the database not authorized to
6. the account ID
7. Attempts to SELECT the list of users and passwords
8. All direct access to the database from accounts which should be limited to access
9. through an application
10. Use of nonstandard tools (E.g. Excel, Access) to directly access DBMS
11. Use of any "utility programs" (E.g. Toad) to now access DBMS
12. Use of the Application ID (ApplID) from a source other than the defined owner
13. Application location (based on hostname or IP address)
14. Log failures, manual logging shut down and attempts to purge
15. Attempts to access OS functionality via the database
12. Known attack profiles, such as Buffer overflow, Denial of Service, SQL injection

Next Steps

1. Conduct threat model assessment for Mojaloop databases
2. Conduct security awareness for Mojaloop developers on database standards, threats and controls
3. Identify security benchmarking tools that can be used to test security configurations for Mojaloop databases
4. Conduct security assessment of existing baseline Mojaloop database configurations and make recommendations as guided by this standard:
    a. Identify vulnerabilities, e.g. patch levels, SQL injection, backup and recovery
    b. Establish audit logging levels configured
5. Develop basic security monitoring dashboards for database activity in Mojaloop using existing infrastructure controls (EFK).

Appendix

STRIDE Model and Controls

Type	**Examples**	**Security Controls**
Spoofing (S)	Threat action aimed to illegally access and use another user's credentials, such as username and password.	Strong authentication, e.g. passwords, MFA, digital signatures
Tampering (T)	Threat action aimed to maliciously change/modify persistent data, such as continuous data in a database, and alter data in transit between two computers over an open network, such as the Internet.	Integrity controls, e.g. digital signatures, permissions (ACLs)
Repudiation (R)	Threat action aimed to perform illegal operations in a system that cannot trace the prohibited operations.	Secure logging and auditing Digital signatures
Information disclosure (I)	Threat action to read a file that one was not granted access to or read data in transit.	Confidentiality controls e.g. permissions (ACLs), encryption
Denial of service (D)	Threat aimed to deny access to valid users by making a web server temporarily unavailable or unusable.	Resilience and business continuity, e.g. permissions (ACLs), Quotas, Filtering
Elevation of privilege (E)	Threat aimed to gain privileged access to resources to gain unauthorized access to information or compromise a system.	Authorization controls e.g. permissions (ACLs), input validation