Kafka and Zookeeper Security Standard

The need for protecting streaming data

Kafka is used in Mojaloop to organize data streams, there is often sensitive data passing through Kafka which needs to be secured. This could be PII, PANs, SSNs, health care records or any other sensitive value.

Main reasons to secure Apache Kafka and Zookeeper:

1. Ensure and maintain compliance from a data protection perspective – keep consumer systems protected by enforcing data protection.
2. Protect sensitive data in the confluent platform (Kafka) environment – reduce risk of data breaches which is a key requirement for regulation such as GDPR, PCI-DSS
3. Reduce risk of distributing sensitive data to unprotected integrated platforms - consumers
4. Enable secure analysis of sensitive data via secure logging and analysis.

Below are best practice recommendations for both Mojaloop OSS and hub operators to consider when implementing their Kafka clusters -

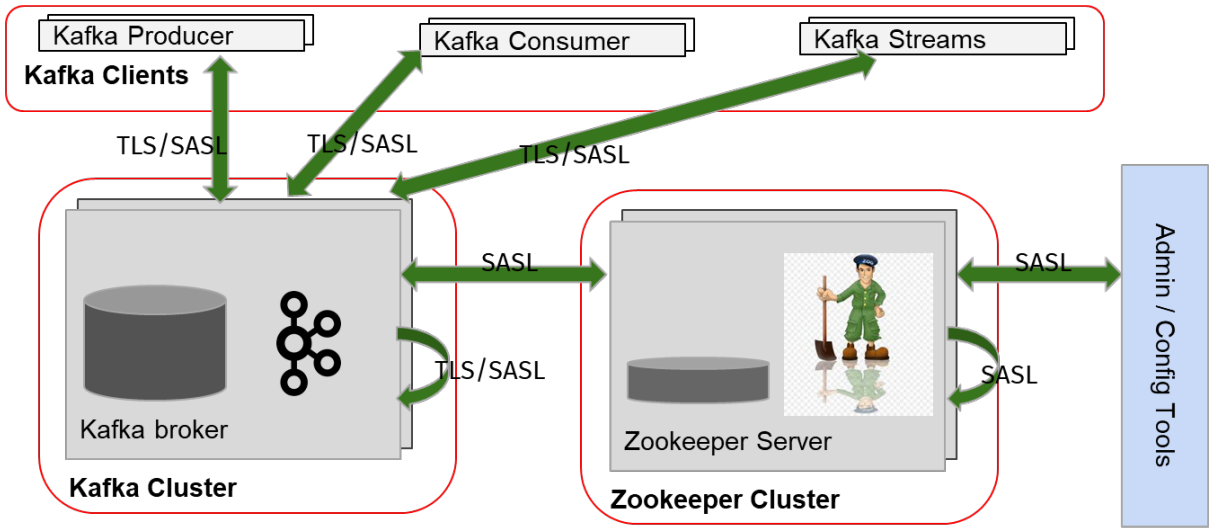
ISSUE	AFFECTS	RISK	RECOMMENDATION
1. Data in transit should not be transmitted in plain text between API gateway and Kafka	Kafka Security	An attacker within the same network as the Kafka cluster may eavesdrop traffic into the cluster resulting in data breach.	Enable encryption of data in-flight using SSL / TLS: This allows your data to be encrypted between your producers and Kafka and your consumers and Kafka.
2. Stream authorization should be put in place for Kafka producers, brokers, and consumers	Kafka Security	An attacker may create rogue producers and consumers and be able to write to or read from unauthorized streams.	Enable authorization rules within Kafka to be able to define which clients can read / write to which topics.
3. Stream authentication not in place for Kafka producers, brokers, and consumers	Kafka Security	An attacker may create rogue producers and consumers and be able to write to or read from unauthorized streams.	Clients need to have and prove their identity before writing to or reading from Kafka streams. Enable authentication using SSL or SASL: This allows your producers and your consumers to authenticate to your Kafka cluster, which verifies their identity before reading from or writing to the cluster.

			We recommend one to choose an authentication mechanism that is already implemented in the enterprise or supported by existing infrastructure.
4. Configuration of quotas within streams	Kafka Security	An attacker may conduct a stream flood and cause a DOS attack by flooding the streams within the Kafka clusters. It is possible for a consumer to consume extremely fast and thus monopolize broker resources as well as cause network saturation. It is also possible for a producer to push extremely large amounts to data thus causing memory pressure and large IO on broker instances.	Currently, the Kafka cluster does not have the ability to throttle/rate limit producers and consumers. It is recommended that a mechanism to enforce quotas on a per-client basis be implemented.
5. Lockdown of Zookeeper access	Zookeeper Security	Restricting access to znodes in Zookeeper can be used to protect Kafka metadata against unauthorized access. Direct manipulation of metadata in Zookeeper is not only dangerous for the health of the cluster, but can also serve as an entry point for malicious users to gain elevated access who can then alter the owner or renewer of delegation tokens.	Restrict access to Zookeeper using network access controls by configuring ACLs to ensure limit access to the Apache ZooKeeper nodes. Kafka ACLs are managed by Zookeeper, hence Zookeeper access needs to be restricted. Configure restrictions to new servers joining the cluster using the zookeeper config file. You can use external authenticators with Zookeeper such as LDAP, AD e.t.c. ACLs can be configured as follows: +Topics – Create, Read, Write, Describe e.t.c +Groups – Read, Write, Describe +Cluster – DescribeConfigs, AlterConfigs, Create
6. Monitoring of Kafka and Zookeeper logs	Kafka Security	A successful attack may go unnoticed of not properly captured in audit logs.	Ensure operational processes around logging management and monitoring (such as Security Operations Center) are in place. Mojaloop implementations currently use Event Framework to log in EFK.

			With authentication and authorization enabled, failed authorizations will appear in INFO logs.
7. Kafka audit logs are not enabled by default should be manually enabled by the administrator	Kafka Security	There is little or no visibility to user and administrative activities around Kafka.	Enable audit logging within Kafka clusters and push logs to the central log server EFK.

Proposed Kafka – Zookeeper architecture

The following is a proposed architecture for a typical Kafka – Zookeeper setup with authentication and encryption enforced:



This design needs to be refined and configurations optimized per environment and factoring in performance implications.

Mojaloop API Touchpoints

The table below highlights how our investigations affect Mojaloop operations and links them to the recommended best practice standards and regulations that may apply to most implementations. Our best practice references were GDPR, PCI-DSS and ISO 27001 for data protection and IT governance frameworks.

Area	Description	Applicability	Components	Source
PII Data	Identification PII data at rest & in-transit	All Mojaloop API except PISP	All DB's and Kafka	Best Practice /GDPR
Log Data	Secure & immutable transaction logs	All data stores and repositories	Mojaloop Core, All DB's and Kafka	Best Practice/ Data Protection
Transaction data	Full call and data flow tracing	All Mojaloop API's except PISP	Mojaloop Core, All DB's and Kafka	Best Practice/ Data Protection
3rd Party Data	Secure 3rd Party data exchange	Party Lookups, Pathfinder, DFSP	Oracle, Mongo DB and Kafka	Best Practice/ Data Protection

Proposed Next Steps

1. Review proposed Kafka security design and align with stakeholders for review and approval – PI 11
2. Build a new environment to test impact of enabling security in Kafka and Zookeeper on Mojaloop operations and performance and Identify operational implications of enabling security in Kafka and Zookeeper – PI 12