BENAICHA Khaoula
AIT KARROUM Khawla
TANANI Kaoutar
4A ICY SI

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE

Compte Rendu TP Kafka Sécurisation SASL

I. [ETUDE] : Présentation des principes de SASL

1- Qu'est ce que c'est SASL?

SASL, fait référence à Simple Authentication and Security Layer, il s'agit d'une couche qui est utilisé des protocoles de communication et met en place un mécanisme d'authentification et de confidentialité dans le but de permettre aux applications de communiquer entre elles de façon sécurisée en s'identifiant mutuellement. Cette identification se fait grâce à l'utilisation d'username et d'un mot de passe.

2- Définition : définir SASL dans un cadre générique

Dans un cadre générique, SASL est une couche de sécurité utilisée dans les protocoles de communication. Cette couche a pour objectif de donner des mécanismes standardisés d'authentification et pour protéger les données.

SASL permet aux applications de s'authentifier de façon mutuelle et de négocier les méthodes d'authentification et de protection afin d'établir une communication qui est sécurisée.

SASL est largement utilisé dans plusieurs protocoles, comme par exemple les protocoles de messagerie électronique (SMTP, IMAP) et le protocole LDAP, tout en offrant une interface extensible dans le but d'implémenter la sécurité dans les applications réseau.

3- Principes de bases ?

Les principes de base de SASL sont les suivants :

a) Authentification :

Fournit des mécanismes d'authentification dans le but de permettre aux applications de vérifier les identités mutuelles des parties impliquées dans cette communication. Il supporte différents mécanismes d'authentification, comme :

- l'authentification basée sur les mots de passe,
- l'authentification à clé publique,
- l'authentification basée sur les certificats, etc.

Pour choisir le mécanisme d'authentification le plus approprié, les clients et les serveurs ont le droit de négocier et choisir en fonction de leurs capacités et de leurs préférences.

b) Négociation :

Permet aux clients et aux serveurs de négocier les paramètres et les options de sécurité avant d'établir la communication en se basant sur leurs préférences. Cette négociation se fait à travers des échanges de messages entre les différentes parties participantes pour atteindre un accord sur les paramètres de sécurité.

c) Confidentialité des données :

Prend en charge la confidentialité des données en chiffrant les informations échangées entre les applications. SASL utilise des algorithmes de chiffrement forts afin de protéger les données sensibles pendant la transmission.

Pour garantir la confidentialité des données, les méthodes de chiffrement telles que SSL/TLS peuvent être utilisées en conjonction avec SASL.

d) Extensibilité :

Conçu pour être extensible, c'est-à-dire il permet de prendre en charge les nouveaux mécanismes d'authentification et de sécurité. Ce qui va permettre aux protocoles de communication d'ajouter des mécanismes personnalisés ou spécifiques à leurs besoins. Cette extensibilité est essentielle afin de s'adapter à l'évolution des exigences de sécurité et permettre une intégration dans divers environnements ainsi que des applications.

II. [ETUDE] : Présentation de SASL appliqué à KAFKA

1. SASL / GSSAPI (Kerberos)
2. SASL / PLAIN
3. SASL / SCRAM-SHA-256 et SASL / SCRAM-SHA-512
4. SASL / OAUTHBEARER

1) ***Les principales caractéristiques de ces quatre principes ci-dessus:***

1. SASL / GSSAPI (Kerberos) :
- Le mécanisme SASL/GSSAPI (Generic Security Services Application Programming Interface) utilise le protocole Kerberos afin d'authentifier et sécuriser.
- Il est basé sur des tickets de sécurité délivrés par un serveur d'authentification centralisé, appelé KDC (Key Distribution Center).
- Ce mécanisme permet une authentification forte basée sur des clés cryptographiques, où les clients et les serveurs établissent une confiance mutuelle à l'aide de tickets.
- Il offre une sécurité robuste et est couramment utilisé dans les environnements d'entreprise où Kerberos est déployé.

2. SASL / PLAIN :
- Le mécanisme SASL/PLAIN est un mécanisme d'authentification simple basé sur des identifiants (nom d'utilisateur et mot de passe).
- Il envoie les informations d'identification en texte brut, ce qui signifie qu'il ne fournit pas de confidentialité des informations d'identification.
- Toutefois, il peut être utilisé efficacement dans des scénarios où la communication est déjà sécurisée, par exemple via une connexion SSL/TLS.
- Il est souvent utilisé dans les environnements de développement et de test.

3. SASL / SCRAM-SHA-256 et SASL / SCRAM-SHA-512 :
- Les mécanismes SASL/SCRAM-SHA-256 et SASL/SCRAM-SHA-512 sont basés sur le mécanisme SCRAM (Salted Challenge Response Authentication Mechanism).

- Ils offrent une méthode sécurisée d'authentification par nom d'utilisateur et mot de passe en utilisant des fonctions de hachage cryptographique.
- SCRAM-SHA-256 utilise l'algorithme de hachage SHA-256, tandis que SCRAM-SHA-512 utilise SHA-512.
- Ces mécanismes offrent une sécurité améliorée par rapport à SASL/PLAIN, car les mots de passe ne sont jamais envoyés en texte brut sur le réseau.
- Ils sont recommandés pour les environnements où une sécurité accrue est nécessaire, justement dans les déploiements en production.

4. SASL / OAUTHBEARER :
- Le mécanisme SASL/OAUTHBEARER autorise l'authentification via le protocole OAuth.
- Il est utilisé dans les scénarios où l'authentification est basée sur des jetons d'accès OAuth générés par un fournisseur d'identité.
- Ce mécanisme est couramment utilisé dans les applications Web et les services API pour permettre aux utilisateurs de s'authentifier à l'aide de leurs identifiants de fournisseur tiers (par exemple, Facebook, Google, etc.).
- Il offre une flexibilité pour intégrer l'authentification dans un écosystème d'authentification plus large utilisant OAuth.

Ces quatre principes SASL offrent différentes méthodes d'authentification et de sécurité pour les communications dans Kafka. Le choix du mécanisme dépend des exigences de sécurité du projet, de la compatibilité avec l'environnement et des fonctionnalités spécifiques qu'on souhaite prendre en charge, comme l'authentification centralisée avec Kerberos ou l'authentification basée sur des jetons OAuth.

2) ***L'apport de chacun de ces principes sur la sécurité d'accès à Kafka :***

Chacun des principes SASL a un apport spécifique sur la sécurité d'accès à Kafka :

1. SASL/GSSAPI (Kerberos) :
- L'apport principal de SASL/GSSAPI réside dans son utilisation du protocole Kerberos, qui permet une authentification forte basée sur des clés cryptographiques.

- Kerberos fournit une infrastructure centralisée d'authentification et de gestion des clés, offrant donc une sécurité renforcée pour l'accès à Kafka.
- L'utilisation de tickets Kerberos assure une confiance mutuelle entre les clients et les serveurs, renforçant ainsi la sécurité des communications.

2. SASL/PLAIN :
- SASL/PLAIN est le mécanisme le plus simple, cependant, il offre tout de même un certain niveau de sécurité en utilisant des identifiants (nom d'utilisateur et mot de passe) pour l'authentification.
- Son apport principal est sa facilité d'utilisation et sa mise en place rapide, ce qui en fait un choix courant pour les environnements de développement et de test.
- Toutefois, il est recommandé de l'utiliser uniquement dans des scénarios où la communication est déjà sécurisée, par exemple via une connexion SSL/TLS, vu qu'il envoie les informations d'identification en texte brut.

3. SASL/SCRAM-SHA-256 et SASL/SCRAM-SHA-512 :
- Les mécanismes SASL/SCRAM-SHA-256 et SASL/SCRAM-SHA-512 offrent une sécurité renforcée par rapport à SASL/PLAIN.
- En utilisant des fonctions de hachage cryptographique (SHA-256 ou SHA-512) et en évitant l'envoi des mots de passe en texte brut, ils protègent les informations d'identification contre les attaques par interception ou par force brute.
- Ces mécanismes sont recommandés pour les déploiements en production où une sécurité accrue est obligatoire, car ils offrent une protection robuste pour l'accès à Kafka.

4. SASL/OAUTHBEARER :
- L'apport principal de SASL/OAUTHBEARER réside dans son intégration avec le protocole OAuth, qui est largement utilisé pour l'authentification et l'autorisation dans les applications Web et les services API.
- Il offrent la possibilité aux utilisateurs de s'authentifier à l'aide de leurs identifiants de fournisseur tiers (par exemple, Facebook, Google) via des jetons d'accès OAuth.

- SASL/OAUTHBEARER permet ainsi une flexibilité pour intégrer l'authentification à Kafka dans un écosystème d'authentification plus large utilisant OAuth.

Globalement, chacun de ces principes SASL apporte des améliorations spécifiques à la sécurité d'accès à Kafka, en fonction des exigences de sécurité, de la complexité de déploiement et des fonctionnalités d'authentification requises dans votre projet.

3) ***Utilité de la sécurisation du Zookeeper :***

Effectivement, il est aussi important de sécuriser ZooKeeper lors de l'utilisation de Kafka. ZooKeeper est un composant central dans l'écosystème de Kafka, qui s'occupe de la coordination, de la gestion des configurations et du suivi de l'état des brokers Kafka.

La sécurisation de ZooKeeper est essentielle afin de garantir la sécurité globale de votre déploiement Kafka. Parmi les raisons pour lesquelles il est recommandé de sécuriser ZooKeeper :

1. Protection des données sensibles : ZooKeeper stocke des informations sensibles, comme les informations d'authentification, les configurations de sécurité et les données de verrouillage. La sécurisation de ZooKeeper garantit que ces données ne sont accessibles qu'aux entités autorisées.

2. Contrôle de l'accès : En sécurisant ZooKeeper, on peut mettre en place des mécanismes d'authentification et d'autorisation pour gérer qui peut accéder à ZooKeeper et effectuer des opérations sur les nœuds et les données.

3. Prévention des attaques : Un ZooKeeper non sécurisé peut être une cible pour les attaques malveillantes, comme la lecture ou la modification non autorisée des données, les attaques par déni de service, ou l'exploitation de vulnérabilités connues. En sécurisant ZooKeeper, on renforce la résilience du cluster Kafka contre ces types d'attaques.

On peut proposer quelques mesures de sécurité qu'on peut prendre pour sécuriser ZooKeeper :

- **Utiliser des connexions sécurisées :** Configurez ZooKeeper afin d'utiliser des connexions chiffrées (SSL/TLS) pour la communication entre les différents nœuds du cluster ZooKeeper.

- Authentification et autorisation : Mettre en place des mécanismes d'authentification solides pour contrôler l'accès à ZooKeeper. On peut utiliser SASL avec les mécanismes d'authentification appropriés, tels que SASL/GSSAPI (Kerberos), SASL/SCRAM-SHA-256, etc. On peut également utiliser des listes de contrôle d'accès (ACL) afin de définir les autorisations d'accès aux nœuds et aux opérations.

- **Configuration appropriée :** Il faut s'assurer que la configuration de ZooKeeper est adaptée à la sécurité. Cela inclut des pratiques telles que la désactivation des fonctionnalités non nécessaires, la limitation des accès, la rotation régulière des informations d'identification, etc.

- **Surveillance et journalisation :** Mettre en place une surveillance et une journalisation appropriées pour détecter les activités suspectes et faciliter les investigations en cas d'incident de sécurité.

En sécurisant ZooKeeper, on contribue à la protection du déploiement Kafka dans son ensemble, renforçant ainsi la sécurité des données et des communications.

4) ***Peut-on se passer de Zookeeper ?***
Son grand remplaçant ?
Le fonctionnement de SASL sur ce remplaçant ? :

Oui, il est possible de se passer de ZooKeeper en se servant d'une alternative appelée **Apache Kafka Streams avec la fonctionnalité KTable**, introduite dans Kafka 0.10.1.0 et les versions ultérieures. Kafka Streams permet de gérer les états et les jointures de manière distribuée sans dépendre de ZooKeeper.

Kafka Streams repose sur les mêmes brokers Kafka utilisés pour le traitement des messages, éliminant donc le besoin d'un composant distinct tel que ZooKeeper pour la coordination et le suivi des états. L'objectif ultime de Kafka Streams est de permettre le traitement en

temps réel et l'analyse des flux de données à l'intérieur du cluster Kafka lui-même.

Quant à la SASL, Kafka Streams prend en charge les mêmes mécanismes d'authentification SASL que Kafka. Autrement dit, on peut configurer Kafka Streams pour utiliser SASL/GSSAPI (Kerberos), SASL/PLAIN, SASL/SCRAM-SHA-256, SASL/SCRAM-SHA-512 ou SASL/OAUTHBEARER afin de sécuriser les communications entre les différents composants du cluster Kafka.

Pour configurer SASL sur Kafka Streams, on doit fournir les paramètres de sécurité appropriés dans les propriétés de configuration de Kafka Streams. Cela peut inclure des informations comme les mécanismes SASL pris en charge, les informations d'identification du client, les chemins vers les fichiers de clés et de certificats, etc.

On donne un exemple de configuration pour utiliser SASL/GSSAPI (Kerberos) avec Kafka Streams :

```properties
# Configuration de SASL pour Kafka Streams
security.protocol=SASL_PLAINTEXT
sasl.mechanism=GSSAPI
sasl.kerberos.service.name=kafka
```

Dans cet exemple, `security.protocol` précise que la communication doit utiliser le protocole **SASL_PLAINTEXT (SASL sur une connexion non chiffrée)**, `sasl.mechanism` spécifie l'utilisation de SASL/GSSAPI et `sasl.kerberos.service.name` indique le nom du service Kerberos associé à Kafka.

Il est important de s’assurer de consulter la documentation spécifique de Kafka Streams et de la plateforme utilisée pour obtenir des informations détaillées sur la configuration de SASL et les mécanismes d'authentification pris en charge.