카프카 인증(AuthN) 인가(AuthZ)의 중요성

2023/08/09

SPITHA Inc.

jwsong@spitha.io

오늘의 내용

kafka KRU SPITHA Felice

- 인증 인가의 중요성
- 인증 구성 : out of the box
  - GSSAPI, PLAIN, SCRAM, **OAUTHBEARER**
- 인가 구성
  - ACL
- Advanced
  - 위임 토큰
  - 사용량 제한

인증 인가의 중요성

Most problems are due to mistakes

kafka KRU SPITHA Felice

- Producer
  - 특정 토픽/파티션에 추가할 메시지를 생성하여 브로커로 전송
  - 관련 없는 토픽에 포맷이 다른 메시지를 추가하면?

- Consumer
  - 특정 토픽/파티션의 메시지를 읽어 감
  - 다른 컨슈머 그룹에 참여하여 메시지를 가져가면?

인증 인가의 중요성

- CLI 도구
  - 누구나 다운로드 받아 사용 가능
  - 토픽/클러스터/클라이언트 설정 조회/변경
  - 토픽 생성/변경/조회/삭제
  - 컨슈머 그룹 오프셋 조회/변경

  - 인증/인가 구성이 없으면 누구든 모든 기능을 사용할 수 있음
  - 컨슈머/프로듀서가 사용하고 있는 토픽을 삭제하면?

인증 인가의 중요성

- AdminClient
  - https://www.javadoc.io/doc/org.apache.kafka/kafka-clients/latest/org/apache/kafka/clients/admin/KafkaAdminClient.html
  - 다양한 관리 함수 제공
    - alterConfigs, alterConsumergroupOffsets, alterPartitionReassignments, alterReplcaLogDirs, ...
    - createTopics, createPartitions, ...
    - deleteConsumerGroupOffsets, deleteConsumerGroups, deleteRecords, deleteTopics, ...

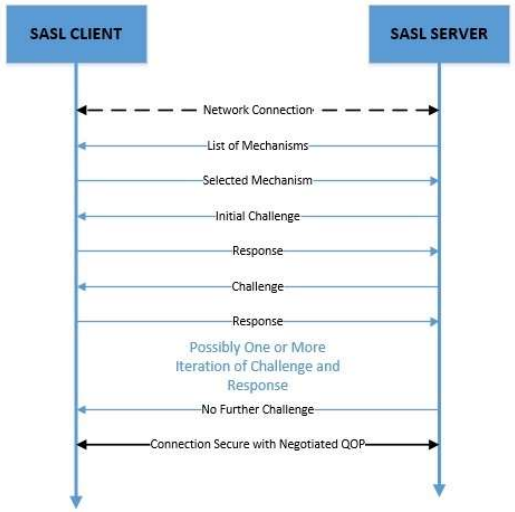
클라이언트 (인증) 프로토콜

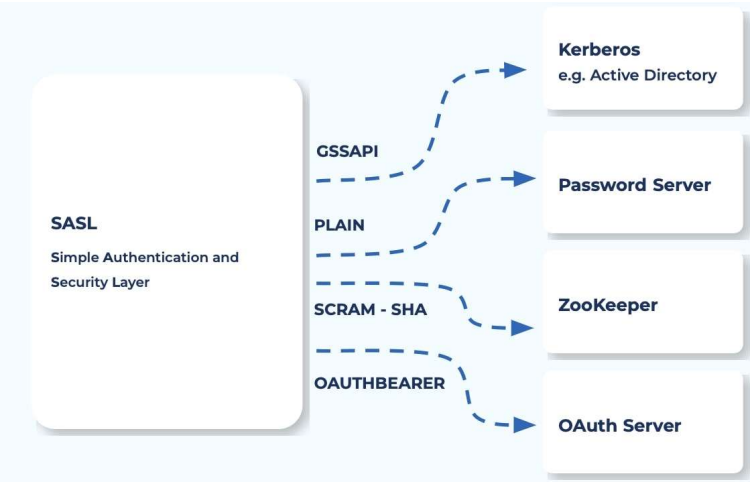
- 인증 없음
  - PLAINTEXT
    - 클라이언트(프로듀서, 컨슈머, 어드민 클라이언트) <-> 브로커간 평문 통신
  - SSL
    - 클라이언트 <-> 브로커간 암호화 통신
- 인증 사용
  - SASL_PLAINTEXT
    - SASL 인증 프로토콜 평문 통신
  - SASL_SSL
    - SASL 인증 프로토콜 암호화 통신

SASL

Simple Authentication and Security Layer

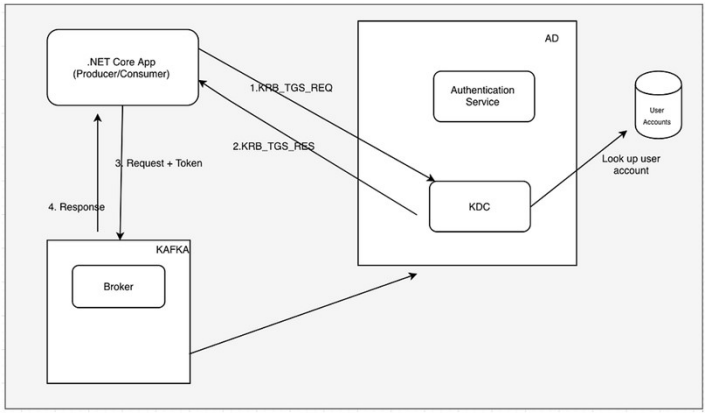
- IETF RFC 4422
  - 클라이언트 서버 간에 인증 메커니즘을 협상하고, 합의된 인증 모듈에 인증 과정을 위임하는 프로토콜





GSSAPI 인증
Kerberos v5

- Generic Security Service API
  - GSSAPI 역시 SASL과 마찬가지로 인증 메커니즘 독립적 API
  - Kafka에서는 Kerberos 인증 메커니즘 지원
    - 계정 정보는 key server에서 관리
      - Keytab file을 배포하기 까다로움
      - 인증이 hostname으로 이루어져 FQDN이 필요함
    - KRB5+OpenLDAP 구성으로 사용 가능
    - MS Active Directory 연동 가능



https://medium.com/tribalscale/kafka-security-configuring-sasl-authentication-on-net-core-apps-da5d0b0fcc5

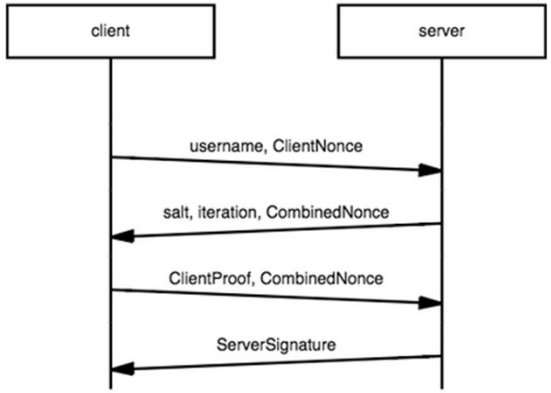
PLAIN 인증 메커니즘
username and password

- SASL/PLAIN
  - username / password로 인증
  - Client 인증 정보를 static file에 보관
    - server.properties 혹은 별도 jaas 파일
  - 인증 정보 변경이 쉽지 않음
    - 인증 정보 변경을 위해
      - 모든 브로커 설정 변경
      - 브로커 재기동

SCRAM 인증 메커니즘

Salted Challenge-Response Authentication Mechanism

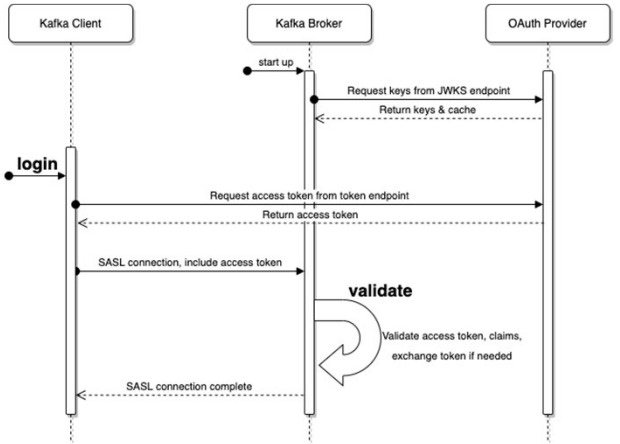
- SCRAM-SHA-256/512
  - Zookeeper / kraft storage에 인증 정보 보관
  - 계정 추가, 삭제, 변경이 용이함
  - 보안성이 확보 되어 있으며 카프카와 가장 잘 통합되어 있음
    - 평문 패스워드 전송 없이 인증
    - CLI 도구 및 API 제공
    - 별도 시스템 없이 인증 기능 제공



https://medium.com/@hussein.joe.au/kafka-authentication-using-sasl-scram-740e55da1fbc

OAUTHBEARER 인증 메커니즘
OIDC on top of oAuth2

- Open ID Connect
  - KIP-768에서 OAUTHBEARER 인증을 확장하여 OIDC 지원
    - Kafka 3.1.0에서 release
    - 이전 버전의 OAUTHBEARER는 reference 구현
  - KeyCloak과 같은 OIDC 지원 인증 서버에서 계정 정보 관리
  - clientcredentials 만 지원하는 것은 단점



https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=186877575

OAUTHBEARER 구성 예

ready to use from version 3.1.0

before version 3.1.0

```
listener.name.sasl_ssl.oauthbearer.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule
required unsecuredLoginStringClaim_sub="admin";

listeners=SASL_SSL://host.name:port (or SASL_PLAINTEXT if non-production)
security.inter.broker.protocol=SASL_SSL (or SASL_PLAINTEXT if non-production)
sasl.mechanism.inter.broker.protocol=OAUTHBEARER
sasl.enabled.mechanisms=OAUTHBEARER
```

https://docs.confluent.io/platform/current/kafka/authentication_sasl/authentication_sasl_oauth.html

from version 3.1.0

```
sasl.enabled.mechanisms=OAUTHBEARER

listener.name.sasl_plaintext.oauthbearer.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginMod
ule required clientId="kafka-broker" clientSecret="client-secret-from-oidc-server";

listener.name.sasl_plaintext.oauthbearer.sasl.server.callback.handler.class=org.apache.kafka.common.security.oauthbearer.s
ecured.OAuthBearerValidatorCallbackHandler

sasl.oauthbearer.jwks.endpoint.url=https://oauth.server/auth/realms/kafka-cluster/protocol/openid-connect/certs

listener.name.sasl_plaintext.oauthbearer.sasl.login.callback.handler=
org.apache.kafka.common.security.oauthbearer.secured.OAuthBearerLoginCallbackHandler

sasl.oauthbearer.token.endpoint.url=https://oauth.server/auth/realms/kafka-cluster/protocol/openid-connect/token
```

매뉴얼 공개

kafka-keycloak OIDC 구성

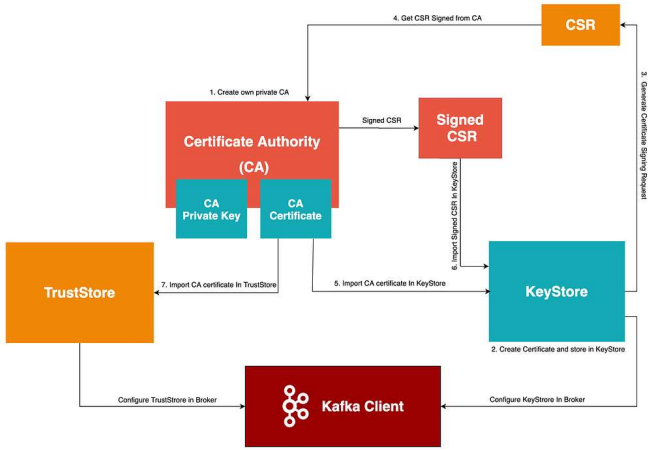
kafka KRU

SPITHA Felice

SSL/TLS or mTLS

- Secure Socket Layer
  - 네트워크를 통한 모든 송수신 데이터를 암호화
  - 인증과는 별도로 분리된 레이어
  - 카프카 브로커용 인증서 필요
  - 암복호화로 인한 CPU 부하 증가
- SASL과 결합하여
  - 여러 인증 메커니즘과 결합하여 사용 가능
  - SASL_SSL/(GSSAPI,PLAIN,...)

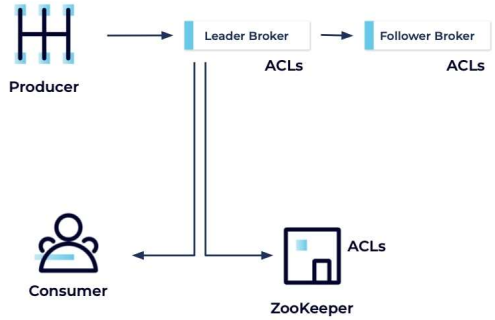
- mutual TLS
  - 양방향 TLS 인증
  - 클라이언트용 인증서 필요
  - 클라이언트 인증서를 서버에서 확인

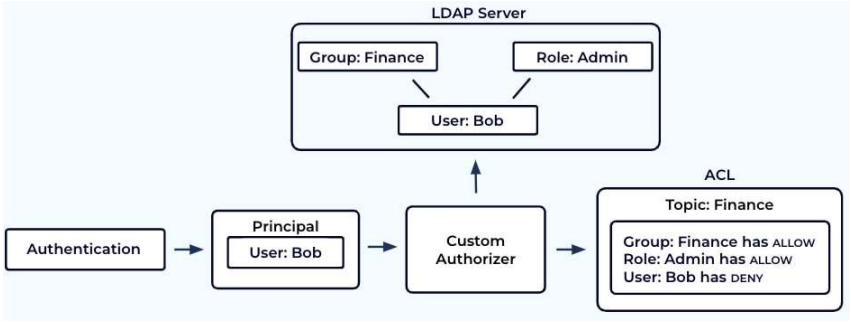


https://medium.com/jinternals/kafka-ssl-client-authentication-part-2-82c211d64eb6

Access Control

- ACL
  - principal **P** is [Allow/Denied]
    - client account
  - operation **O**
    - Read, Write, Create, Delete, Alter, Describe, ...
  - from host **H**
  - on any resource **R** / matching resourcepattern **RP**
    - Cluster, Topic, Consumer group, Transactional ID
- AuthN이 구성되어 있어야 함

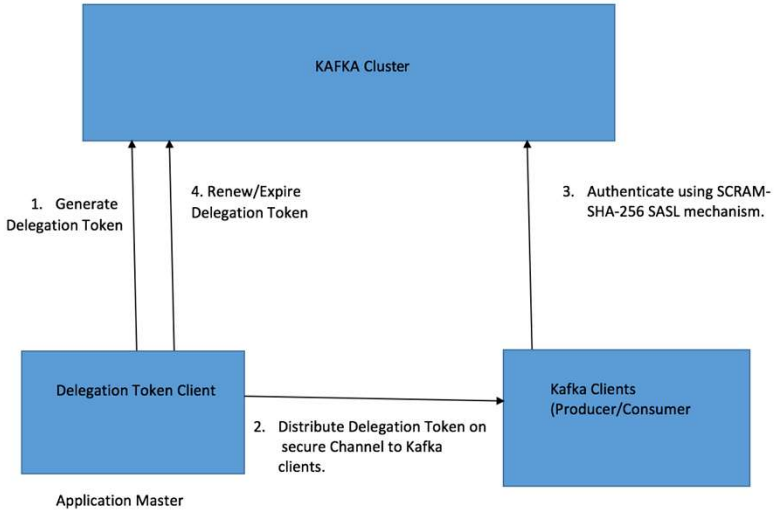




https://developer.confluent.io/courses/security/authorization/

Delegation Token

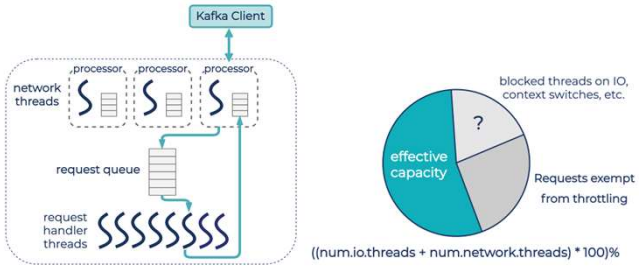
- 외부 시스템 의존 인증 방법은
  - Krb5, mTLS, OAuth2
  - 인증 관련 파일 배포 어려움
    - keytab, keystore
  - Ticket 혹은 token을 주기적으로 refresh 해야 함
    - 외부 인증 시스템에 부하가 몰릴 수 있음 → 시스템 지연
    - 외부 인증 시스템 장애 → 프로듀싱/컨슈밍 장애
- 이를 대체하기 위해
  - 만료 기간을 갖는 위임 토큰을
  - 브로커에서 생성하여
  - 주키퍼에 보관하고
  - SCRAM 메커니즘으로 인증
- 발급/갱신/폐기가 간편



사용량 제어

one rotten apple spoils the whole barrel

- Throttling
  - 전체 서비스 품질(QoS)를 보장하기 위해 특정 작업 혹은 클라이언트의 대역폭 등을 제한할 필요가 있음
    - 특정 시간에 대용량 트래픽을 발생시킬 가능성이 있는 프로듀서
    - 브로커간 디스크 사용량 균형을 유지하기 위한 파티션 재배치 작업
    - 기타



https://www.confluent.io/blog/cloud-native-multi-tenant-kafka-with-confluent-cloud/

- Client Quotas
  - 네트워크 대역폭 할당량
    - 바이트 전송율 임계값
  - 요청 속도 할당량
    - 네트워크 쓰레드, I/O 쓰레드의 CPU 사용율 임계값
  - (user, client-id), user 혹은 client-id에 할당
    - 전체 클라이언트 그룹에서 할당량 공유

Questions?