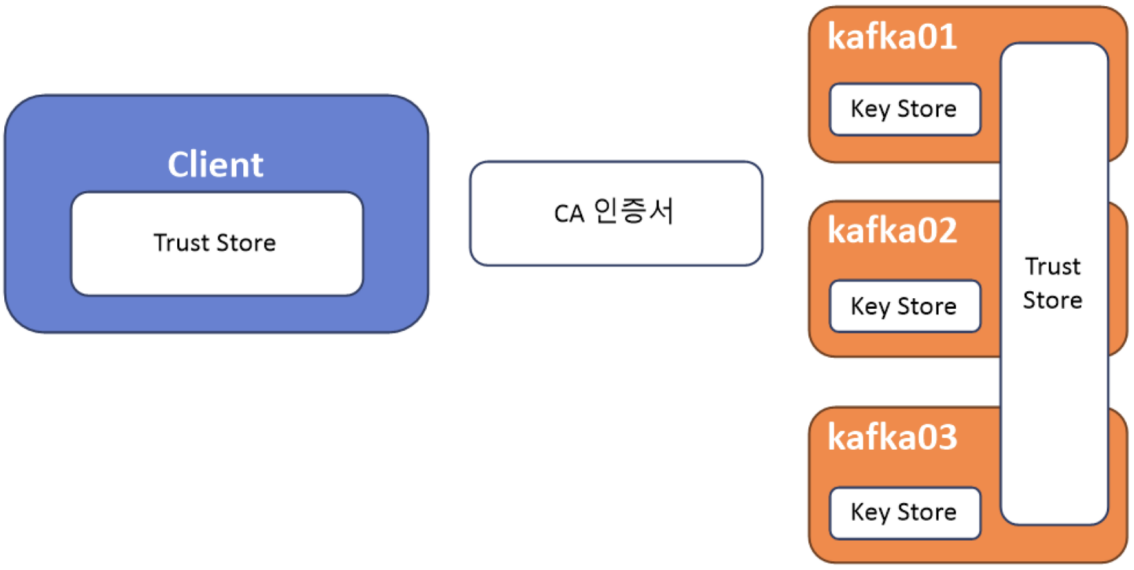
9장

9.1 카프카 보안의 세가지 요소

- 암호화, 인증, 권한

1. 암호화 (SSL)
    - SSL은 통신 보안을 적용하기 위한 표준 암호 규약
    - `대칭키 방식`은 오버헤드가 적은 반면 `키 노출 위험성이 존재`하고 `비대칭키 방식`은 `오버헤드`가 큽니다.
2. 인증(SASL)
    - SASL/GSSAPI
        - 커버로스 인증 방식, 회사내부에 커버로스 서버(kerberos)가 있다면 사용
        - `티켓 기반의 컴퓨터 네트워크 인증 프로토콜`
    - SASL/PLAIN
        - 아이디/비밀번호를 텍스트 형태로 사용
        - 개발환경에서 사용
    - SASL/SCRAM-SHA-256, SASL/SCRAM-SHA-512
        - 암호에 해시된 내용 추가
        - 솔티드 챌린지 응답 인증 메커니즘
    - SASL/OAUTHBEARER
        - 3.1.0 릴리즈부터는 별도의 핸들러 필요없이 사용 가능
3. 권한(ACL)
    - 규칙기반의 리스트를 만들어 접근을 제한, 주피커에 저장
    - 리소스 타입(토픽, 그룹, 클러스터, 트랜잭셔널 ID, 위임 토큰)별로 구체적인 설정 가능

9.2 SSL을 이용한 카프카 암호화



카프카 SSL 적용을 위한 전반적인 구성도 (클러스트 환경)

- 자바 기반 애플리케이션에서는 키스토어(KeyStore) 인터페이스를 통해 퍼블릭 키, 프라이빗 키, 인증서를 추상화해 제공
- 카프카는 `keytool`이라는 명령어로 카프카에 SSL 적용 작업을 진행

9.2.1 브로커 키스토어 생성

- 각 브로커 마다 프라이빗 키와 인증서를 저장하는 키스토어 생성
- `KeyStore` vs `TrustStore`
  - `KeyStore` : 프라이빗 키와 인증서 저장하고 자격 증명을 제공 + 프라이빗한 정보 저장
  - `TrustStore` : 퍼블릭 키와 유효성을 검사하는 서명된 인증서 저장 + 프라이빗한 정보 저장 X

```
keytool -keystore kafka.server.keystore.jks -alias localho
keytool -list -v -keystore kafka.server.keystore.jks
```

9.2.2 CA 인증서 생성

```
openssl req -new -x509 -keyout ca-key -out ca-cert -days 356
```

소스의 프라이빗 키 비밀번호

9.2.3 트러스트스토어 생성

- 자체 생성한 CA 인증서를 클라이언트가 신뢰할 수 있도록 트러스트스토어에 추가

```
keytool -keystore kafka.server.truststore.jks -alias CARoot -
keytool -list -v -keystore kafka.server.truststore.jks
```

keytool	키스토어 이름	file	인증서 파일
alias	별칭	storepass	저장소 비밀번호
importcert	인증서를 임포트	keypass	키 비밀번호

9.2.4 인증서 서명

```
keytool -keystore kafka.server.keystore.jks -alias localhost
#서명 적용
openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -ou
# 키스토어에 인증서 파일인 ca-cert와 서명된 cert-signed 추가
keytool -keystore kafka.server.keystore.jks -alias CARoot -im
keytool -keystore kafka.server.keystore.jks -alias localhost

# 키스토어 내용 확인
keytool -list -v -keystore kafka.server.keystore.jks
```

x509	표준 인증서 번호	in	인풋 파일
req	인증서 서명 요청	out	아웃풋 파일
ca	인증서 파일	days	유효 일자
cakey	프라이빗 키파일	passin	소스의 프라이빗 키 비밀번호

소스의 프라이빗 키 비밀번호

x509	표준 인증서 번호	days	유효일자
new	새로 생성 요청	subj	인증서 제목
keyout	생성한 키 파일 이름	nodes	프라이빗 키 암호화 X
out	생성한 인증서 파일 이름		

- 모든 브로커에 동일한 설정 필요

```
mkdir ssl
expert SSLPASS=peterpass
```

```
keytool -keystore kafka.server.keystore.jks -alias localhost

mkdir ssl
expert SSLPASS=peterpass
keytool -keystore kafka.server.keystore.jks -alias localhost
```

- kafka1에서 생성한 CA인증서와 키파일 복사

```
ssh-keygen
cat /root/.ssh/id_rsa
# 출력값
MIIEpAIBAAKCAQEApMopPFjJViwuyJP+s8U4ogKLsxyjHCTG23HxjaVxIq
9m4ZrUO//DCSaoVv+RFMoR9Nnr6j2Ml+2i58dS2GMXIu7sGL892SxG9GyE
HlwjSQJSG1zEH8aqPyFp+HoJhirc+L8BHe1ziZPnM2XR8oQNq6+nlPrtvD
eCuoWA2TSt8lDGn9a3LNlRfnAH3b7ILT+3nlYgd51wSJ5HaGuGW1GW5TsQ
LfhhZJHjgB+2rEK/x5hGy84QUmJXT1wHsaWDRlXs0+6iqRKzPACQRTN/En
SGz4QqnhJw1TmvbAqjbIaeuAC3P8Qc+PK1VPSwIDAQABAoIBADmUulvv2k
Y8uFOojBwvbPJt+KqkG7yt/n02R8mHN2SpWZWzeD2O3SOxclkT7CoLgcLA
H/E1FSAMzDsy3ttmnfSUk9g32c/Yz2+WrtZk859rRC1zeMRxssNkdZbxba
Z1yBwEFugPcguP+4zN4U/sW7+J4uFxcYfVED3jUHqm+YjMDLSieD9WRJC4
An+Q5CFusdr3TNWJWzBOTqqzxKWiQOmv9eUte6ayqxIGHX1TRf5hq9hkKF
rS9qDtISU0rFs9xQth2tfhGvDwLUz4dbLmH3qkHLtzcucj8xN0vMQReAcI
y/8hbcECgYEAzWbdDSYDOnF5sK4Qdfh+XG0SJpBnvsXz2JfsNoqlALtu8n
+gkQEaHIGEHBNdlZBSjLhwPuq9K/QuIrDjqkX9q2rk8lplfBwTZn+G5Tgw
dFdxnADxI/cxpgm3wtdRnT6A0KyYN9DnO/MVwKI8rjimyd5IhJjjFAMCgY
5/NvlNajRgZZTNJG0R8hgMiEhDUr8jGxjECLhHC52aVK12aWnFElRGO5Mv
J6/RquMMdaVnbvdhqWTNJy0CzjMH3kUthbKm9URMlQ1I0t+2+vE3q8p8CC
ubqY6n57VK7NLvRHV1EuY/zJ32xCQW6iVv4myRkCgYA/aOoFDR8iA7YVnS
AY4A6cOdWW8WBpCdpd+agkKWClfpfh4zKmkTiGZFVsdW4Gpi9V0fqU8dLE
SYNFNDgBR93PpHfkNe+S+jw9zS+ZgyEnwXcJOiE3gTCKmaVpOR+w84LA6Q
+PRWK7+f6U1UNJMenDQ+EQKBgQDKanOdYrCf0gJMq0krizo8NCBC2lywJ+
wDhAqRChzmQ6cmMRYHAIPPhzoPAQpxU0i/VqJCdNc1zDYpAf8La4W4YeTv
pPYjV9LaO6OAIPgnzcB+USt4rMsgSlVgAOVghpNqXHTgR+RfzCSaarNSvV
uHPlUQKBgQC8lAXSSHZ6As454b/+FBS/K/HmNqGDbhzXSFmWZrH0a3lHVB
AFAHk5T4d3xN/vgGMPbE9y38bDT5m7xwvj/snuhSf1/cA1uU5TE3McS7D4
g7WPaOmTJyEhC40B4wat5V6Rwm6SJaKT1lHGq58A0upL/NK2R5TFAQ==
```

  - authorized_keys 파일을 찾지 못함

9.3 커버로스(SASL)를 이용한 카프카 인증

- 티켓을 기반으로 하는 컴퓨터 네트워크 인증 프로토콜, 사용자이 신원을 식별
- 하나의 커버로스 티켓을 이용해 여러 서비스에서 사용가능 (싱글 사인온)
- 티켓 인증
  - 카프카 클라이언트가 인증/티켓 서버인 커버로스 서버로 인증요청
  - 커버로스 서버는 카프카 클라이언트로 티켓 발급
  - 카프카 클라이언트는 이 티켓으로 카프카 서버에 인증

9.3.1 커버로스 구성

```
# 커버로스 설치

# Principal 객체 형성
kadmin.local -q "add_principal -randkey peter01@FOO.BAR"
kadmin.local -q "add_principal -randkey peter02@FOO.BAR"
kadmin.local -q "add_principal -randkey peter03@FOO.BAR"

# 서비스를 위한 Principal 생성
kadmin.local -q "add_principal -randkey kafka/kafka1@FOO.BAR"
kadmin.local -q "add_principal -randkey kafka/kafka2@FOO.BAR"
kadmin.local -q "add_principal -randkey kafka/kafka3@FOO.BAR"

# 키탭 생성 (별도 비밀번호 필요 X)
mkdir -p /var/lib/kafka/keytabs
kadmin.local -q "ktadd -k /var/lib/kafka/keytabs/peter01.user
kadmin.local -q "ktadd -k /var/lib/kafka/keytabs/peter02.user
kadmin.local -q "ktadd -k /var/lib/kafka/keytabs/peter03.user
```

9.3.2 키탭을 이용한 인증

```
# kafka1에 만든 Keytab 다른 브로커에 복사 필요
```

9.3.3 브로커 커버로스 설정

```
# server.properties 변경 -> /ect/kafka/server.properties

# jaas.conf 파일생성 : java 사용자 인증에 대한 표준 API
```

```
# 모든 브로커에 설정
```

9.3.4 클라이언트 커버로스 설정

```
# kafka_client_jaas.conf 파일 생성

# keberos.config 파일 생성
```

9.4 ACL(Access Control List)을 이용한 권한 설정

- 유저별로 특정 토픽에 대한 접근을 허용할지의 여부 설정

9.4.1 브로커 권한 설정 (모든 브로커에 설정정)

```
#server.properties 추가
authorizer.class.name=kafka.security.authorizer.AclAuthorizer
super.users=User:admin;User:kafka
#admin과 kafka가 슈퍼유저
#기본정책은 차단 정책

#환경변수 초기화
unset KAFKA_OPTS
#새로운 토픽 생성
kafka-topics --zookeeper zk1:12181 --create --topic peter-tes
kafka-topics --zookeeper zk1:12181 --create --topic peter-tes

#유저 권한 설정 (토픽에 대한 규칙 추가)
# peter01 :peter-test09 읽기/쓰기
kafka-acls --authorizer-properties zookeeper.connect=zk1:1218
# peter02 : peter-test10 읽기/쓰기
kafka-acls --authorizer-properties zookeeper.connect=zk1:1218

#현재 적용 되는 ACL 규칙 리스트 확인
kafka-acls --authorizer-properties zookeeper.connect=zk1:1218
#컨슈머 그룹에 대한 ACL 규칙 추가
kafka-acls --authorizer-properties zookeeper.connect=zk1:1218

#특정 유저로 티켓 발급
```

```
kinit -kt /var/lib/kafka/keytabs/peter1.user.keytab
export KAFKA_OPTS="-Djava.security.auth.login.config={{/etc/k
```