MQ Broker 안전 사용 가이드

Kafka 보안 설정 가이드

작성자	BoB 13th
작성일	2024.12.21

목차

1. 개요

1.1. 배경

Kafka 는 2011 년 LinkedIn 에서 처음으로 개발된 Message Queue Software 이다. 대량의 데이터를 실시간으로 처리하고 전송하는 데 최적화되어 수많은 이벤트와 메시지를 안정적이고 빠르게 처리할 수 있는 강력한 플랫폼이다. 또한 분산 아키텍처 덕분에 높은 확장성으로 시스템을 사용할 수 있으며 고가용성 및 내결함성을 제공한다.

현재는 Apache Software Foundation 의 오픈소스 프로젝트로 발전하여 전 세계에서 메시지 큐, 로그 관리, 데이터 파이프 라인, 실시간 분석 등 다양한 용도로 활용되고 있다. Netflix, Uber, Spotify, Airbnb, Goldman, Slack 등 세계적인 해외 기업부터 네이버, 우아한 형제들, 쿠팡, Toss 등의 국내 유명 기업들까지 Kafka 시스템을 적극 활용 중에 있다. 이렇듯 Kafka 는 여러 기업들과 산업군에서 효율적인 데이터 흐름 관리와 분석을 위해 필수적인 인프라로 자리매김하고 있다.

국내외에서 많은 기업들이 Kafka 를 사용하는 만큼, 보안 위험이 불러오는 파급력 또한 매우 크다. Kafka 는 사용자에게 수백 가지의 설정 옵션을 제공하며, 이러한 설정들이 시스템의 성능과 보안에 큰 영향을 미친다. 하지만 Kafka 는 기본적으로 보안 설정이 적용되지 않은 상태로 제공되기 때문에, 동일한 네트워크 대역 내의 클라이언트라면 누구든 자유롭게 접근할 수 있다. 이로 인해 악의적인 사용자가 주요 정보에 손쉽게 접근하거나 유출될 위험이 존재한다. 또한 시스템의 유연성을 극대화하고 성능을 최적화하는 과정에서 이러한 설정들을 잘못 조작하게 된다면 리소스 고갈, 데이터 유출, 권한 우회, 서비스 거부 공격 등과 같은 다양한 보안 사고로 이어질 수 있다. 따라서 Kafka 사용자들은 이러한 Kafka 의 보안 설정들을 정확히 이해하고 적절히 관리하는 것이 필수적이다.

본 가이드는 Kafka 를 안전하게 운영하고 발생 가능한 보안 위협을 최소화하기 위한 핵심 보안 설정을 다룬다. Kafka 내부에서 제공하는 다양한 보안 관련 설정 항목을 중점적으로 다루며 안정적으로 Kafka 클러스터를 운영하기 위한 적절한 설정 방법과 권장 사항을 제공한다.

1.2. 진단 List

본 가이드는 Kafka 보안 점검을 위한 진단 리스트를 기반으로 작성되었다. Broker, Producer, Consumer 등 다양한 리소스별로 진단 항목을 제시하며, SASL 과 SSL 설정법 및 이에 대한 진단 항목도 포함하고 있다. 각 항목에서는 발생 가능한 보안 위험, 설정 방법, 그리고 적절한 설정 적용을 위한 판단 기준 등을 제공한다. 이를 통해 사용자는 각 진단 항목을 단계적으로 검토하고 적용하여, 시스템을 안전하게 운영할 수 있도록 지원한다. 아래 표는 전체 진단 리스트를 정리한 내용이다.

분류	소분류	점검 항목
1. Client	1.1	Produce 메시지 크기
	1.2	Produce 메시지 타임아웃
	1.3	Consumer 메시지 offset
	1.4	Consumer offset auto commit
	1.5	요청 송/수신 버퍼
	1.6	소켓 연결 타임아웃
	1.7	클라이언트 DNS 조회
	1.8	메시지 전송 보장
2. Broker	2.1	통신 프로토콜
	2.2	소켓 통신
	2.3	메타데이터 관리
	2.4	연결 복구
	2.5	설정 변경 정책
	2.6	브로커 허용 최대 메시지 크기
	2.7	브로커 간 하트비트
3. Topic	3.1	내부 토픽 접근 관리
	3.2	토픽 자동 생성
	3.3	레코드 배치 크기
	3.4	토픽 생성 정책
4. ACL	4.1	최소 권한 원칙
	4.2	Wildcard
	4.3	Super User
	4.4	Cluster 리소스 관리
	4.5	Describe 권한 관리
	4.6	불필요한 계정 관리

	4.7	ACL 미적용 리소스 접근 허용
5. SSL	5.1	클라이언트 인증 요구 설정
	5.2	SSL 프로토콜
	5.3	프로토콜 매핑
	5.4	SSL 인증서 관리
6. SASL	6.1	암호화 통신
	6.2	토큰 갱신
	6.3	SASL 인증 메시지 크기
	6.4	SASL 로그인 처리 시간 및 횟수

2. Client

2.1. 개요

2 장에서는 Kafka 에서 메시지를 송수신하는 핵심 역할을 담당하는 클라이언트에 대해 다룬다. 클라이언트는 Producer 와 Consumer 를 포함하며, 데이터를 주고받는 과정에서 Kafka 브로커와 직접 통신하기 때문에 보안에 특히 신경을 써야 한다. 클라이언트는 설계와 설정이 잘못될 경우, 메시지 손실, 데이터 누출, 성능 저하와 같은 문제가 발생할 수 있으며, 이는 Kafka 전체 시스템에 안정성을 저해할 수 있다.

따라서 이 장에서는 Kafka 클라이언트 구성 시 반드시 고려해야 할 보안 및 성능 관련 진단 항목을 소개한다.

2.2. Client 진단 List

분류	소분류	점검 항목
Client	1.1	Produce 메시지 크기
	1.2	Produce 메시지 타임아웃
	1.3	Consumer 메시지 offset
	1.4	Consumer offset auto commit
	1.5	요청 송/수신 버퍼
	1.6	소켓 연결 타임아웃
	1.7	클라이언트 DNS 조회
	1.8	메시지 전송 보장

2.3 Client 진단 항목

<table>
<tr><th>No 1.1</th><th>Produce 메시지 크기</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 클라이언트가 브로커로 메시지를 송신할 때의 메시지 크기 관련 설정의 적정성 점검</td></tr>
<tr><td>보안위협</td><td>■ 과도한 메시지 크기 전송 시 시스템 리소스 고갈 및 서비스 중단 위험 및 데이터 처리 실패 발생 가능
■ 지나치게 작은 배치 크기 및 메시지 제한 설정으로 전송 효율 저하</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Producer client, broker, kafka/config/producer.properties
■ max.request.size(default: 1048576) : 프로듀서가 브로커에 한 번 요청으로 보낼 수 있는 최대 크기
■ buffer.memory(default: 33554432) : 프로듀서가 서버로 전송 대기 중인 레코드를 버퍼링하는 데 사용할 수 있는 총 메모리 크기
■ batch.size(default : 16384) : 프로듀서가 메시지를 배치로 처리할 때 배치 하나당 최대 크기
■ message.max.bytes(default: 1048588) : 단일 배치의 최대 크기</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 크기 제한 설정값이 시스템 성능, 평균 메시지 크기에 적합하게 구성
■ batch.size와 linger.ms가 평균 메시지 크기 및 네트워크 대역폭에 적합한 효율로 설정된 상태</td></tr>
<tr><td>취약
■ 과도한 메시지 크기로 설정되어 리소스 고갈 가능성이 높거나, 지나치게 작아 데이터 처리 실패가 일어날 수 있는 상태
■ buffer.memory와 메시지 크기 설정 불일치로 리소스 초과 발생 가능한 상태
■ 지나치게 작은 batch.size로 네트워크 병목 현상이 유발할 수 있는 상태.</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ max.request.size와 message.max.bytes를 일관성 있게 설정
■ buffer.memory를 브로커 용량에 맞게 조정하여 과도한 리소스 점유 방지
■ batch.size를 적절히 설정하여 효율적 전송 유지</td></tr>
</table>

■ 메시지 크기에 맞춘 linger.ms 값 조정

설정 예시

■ kafka/config/producer.properties에 아래 예시와 같이 설정

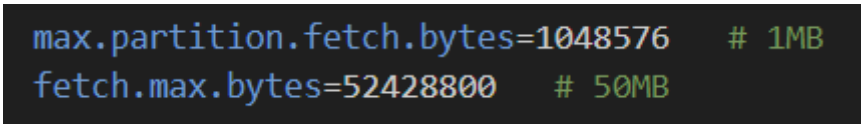
```
max.request.size=5242880    # 5MB
message.max.bytes=5242880   # 5MB
fetch.max.bytes=10485760   # 10MB
max.partition.fetch.bytes=5242880   # 5MB
batch.size=32768    # 32KB
buffer.memory=67108864    # 64MB
```

<table>
<tr><th>No 1.2</th><th>Produce 메시지 타임아웃</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 프로듀서 타임아웃 설정의 적정성 점검</td></tr>
<tr><td>보안위협</td><td>■ 타임아웃 값이 지나치게 짧거나 길 경우 지연 발생 가능성
■ 타임아웃 설정이 부적절할 경우 네트워크 병목 현상이나 브로커의 과부하로 인해 서비스 응답 시간 증가 가능성
■ 암호화가 설정되지 않은 경우, 장시간의 타임아웃은 데이터 탈취 위험 증가</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Producer client, kafka/config/producer.properties
■ delivery.timeout.ms(default: 120000) : 메시지 요청 메소드인 send() 호출이 반환된 후 성공 또는 실패를 보고하는데 최대 시간. 레코드가 지연되는 총 시간, 브로커의 확인을 기다리는 시간, 재시도 가능한 전송 실패에 대해 허용되는 시간 등을 제한
■ linger.ms(default: 0) : 레코드를 배치로 묶어 보내기 위해 대기하는 시간</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ delivery.timeout.ms is greater than request.time.ms + linger.ms
■ Set appropriate timeout based on average network response time and broker throughput
■ Timeout values and encryption communication settings combine to maintain security</td></tr>
<tr><td>취약
■ delivery.timeout.ms 가 request.time.ms + linger.ms 보다 큰 상태
■ 평균 네트워크 응답 시간과 브로커 처리량에 기반한 적절한 타임아웃 설정
■ 타임아웃 값과 암호화 통신 설정이 병행되어 보안성 유지</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ delivery.timeout.ms 값을 네트워크 환경, 브로커 처리 시간, 메시지 크기에 맞게 적절히 설정 설정 일관성 유지
■ linger.ms와 request.timeout.ms의 설정값을 적절히 고려하여 일관성 유지

설정 예시
■ kafka/config/producer.properties에 아래 예시와 같이 설정</td></tr>
</table>

```
delivery.timeout.ms=60000  # 60초 (1분)
linger.ms=10  # 10ms
```

<table>
<tr><th>No 1.3</th><th>Consumer 메시지 offset</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ Consumer의 데이터 요청 크기를 제어하는 주요 설정 값 점검</td></tr>
<tr><td>보안위협</td><td>■ 설정 값이 지나치게 높으면 악의적인 소비자가 과도한 데이터를 요청하여 브로커의 리소스를 소모시킬 수 있음.
■ 설정 값이 비효율적일 경우, 불필요한 네트워크 요청 증가로 인해 처리 속도가 저하될 수 있음.</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Consumer client, kafka/config/consumer.properties
■ max.partition.fetch.bytes(default: 1048576) : 파티션별로 반환할 수 있는 최대 데이터 크기
■ fetch.max.bytes(default: 52428800) : 소비자의 데이터 요청에 대해 반환할 최대 데이터 크기</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 소비자의 메모리 용량과 시스템 요구에 맞게 적절히 설정된 경우
■ 브로커(message.max.bytes)와 토픽(max.message.bytes) 설정 값 내에서 운영</td></tr>
<tr><td>취약
■ 지나치게 높거나 낮게 설정하여 데이터 처리 실패 또는 리소스 소모 발생
■ 설정 값이 높아 메모리 초과로 인한 애플리케이션 장애 발생 가능</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ max.message.bytes 와 message.max.bytes 를 일관성 있게 설정
■ 소비자 애플리케이션의 메모리 용량을 고려하여 메모리 부족 현상 방지

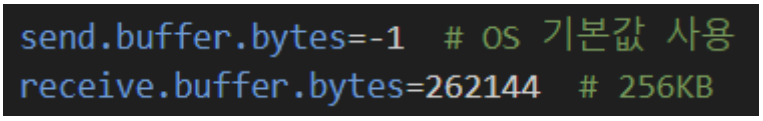
설정 예시
■ kafka/config/consumer.properties 에 아래 예시와 같이 설정
 
</td></tr>
</table>

<table>
<tr><th>No 1.4</th><th>Consumer offset auto commit</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 소비자가 메시지를 처리하고 오프셋을 관리하는 주요 파라미터 점검</td></tr>
<tr><td>보안위협</td><td>■ 커밋 간격이 너무 길 경우, 애플리케이션 장애 시 커밋되지 않은 오프셋 이후 메시지가 중복 소비되거나 손실될 가능성 존재
■ 자동 커밋은 수동 커밋에 비해 보안 제어가 약하기 때문에 악의적인 소비자가 데이터 처리 순서를 왜곡할 가능성 존재</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Consumer client, kafka/config/consumer.properties
■ enable.auto.commit(default: true) : 오프셋을 자동으로 커밋할지 여부 설정
■ auto.commit.interval.ms(default: 5000) : 자동 오프셋 커밋 간격 설정
■ auto.offset.reset(default: latest) : 오프셋이 없을 경우 메시지를 읽을 시작 위치 설정</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ auto.commit.interval.ms 가 적절한 간격으로 설정되어 있는 상태
■ auto.offset.reset : 중요한 데이터 처리에 earliest 또는 none 이 적용된 상태
■ enable.auto.commit : false 로 수동 커밋을 통해 세밀하게 제어하는 상태</td></tr>
<tr><td>취약
■ auto.commit.interval.ms 가 너무 길거나 짧아 성능 저하 및 데이터 손실 발생 가능
■ 중요한 데이터 처리에 auto.offset.reset 이 latest 로 설정된 경우</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ auto.commit.interval.ms 가 너무 긴 간격으로 설정되어 데이터 손실이나 중복 처리가 발생하지 않도록 소비자의 장애나 네트워크 지연을 고려해 적절한 값 설정
■ 주요 데이터를 처리할 때는 earliest 나 none 값을 사용하여 메시지 손실 방지
■ latest 는 실시간 데이터 처리에만 사용하는 것을 권장

설정 예시
■ kafka/config/consumer.properties 에서 아래 예시와 같이 작성</td></tr>
</table>

```
enable.auto.commit=false
auto.commit.interval.ms=10000  # 10초
auto.offset.reset=earliest
```

<table>
<tr><th>No 1.5</th><th>요청 송/수신 버퍼</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 클라이언트의 송/수신할 때 사용할 TCP 버퍼 관리</td></tr>
<tr><td>보안위협</td><td>■ 버퍼 크기 설정이 적절하지 않은 경우 과도한 시스템 리소스 소비로 인해 서비스 거부(DoS) 공격에 취약할 수 있음
■ 너무 작은 버퍼는 대량의 메시지가 빠르게 도착할 때 데이터 손실을 유발할 수 있음</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Producer client, Consumer client, kafka/config/producer.properties, kafka/config/consumer.properties
■ send.buffer.bytes (default : 131072) : 데이터를 전송할 때 사용할 TCP 송신 버퍼의 크기
■ receive.buffer.bytes (default : 32768) : 데이터를 읽을 때 사용할 TCP 수신 버퍼의 크기</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 송/수신 버퍼의 크기가 적절하게 설정되어 있으며 리소스가 과도하게 소모되지 않고 성능 저하나 데이터 손실 없이 원활하게 작동하는 상태</td></tr>
<tr><td>취약
■ 버퍼의 크기가 너무 작거나 너무 커서 데이터 손실, 성능 저하, 리소스 부족 등의 문제가 발생할 가능성이 있는 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ 시스템의 네트워크 환경과 트래픽 패턴에 맞춰 receive.buffer.bytes 와 send.buffer.bytes 설정
■ 해당 설정 값을 -1 로 설정할 경우 OS 기본값이 적용
■ OS 기본값이 최적화 되어있고, 시스템 네트워크 환경을 고려한 경우 해당 설정 반영

설정 예시
■ kafka/config/consumer.properties 와 kafka/config/producer.properties 에 아래 예시와 같이 작성
 
</td></tr>
</table>

<table>
<tr><th>No 1.6</th><th>소켓 연결 타임아웃</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 클라이언트가 브로커와 TCP 연결을 설정할 때 필요한 타임아웃 시간 설정</td></tr>
<tr><td>보안위협</td><td>■ 타임아웃 설정이 비효율적일 경우, 비정상적인 접근 차단에 실패할 가능성 존재
■ 비정상적인 연결 요청에 대해 과도하게 대기하면 서비스가 리소스를 소모하여 거부 공격을 유발할 가능성 존재</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Producer client, Consumer client, kafka/config/producer.properties, kafka/config/consumer.properties
■ socket.connection.setup.timeout.max.ms(default: 30000) : 프로듀서가 브로커 연결을 시도할 때 대기하는 시간
■ socket.connection.setup.timeout.ms(default: 10000) : 클라이언트가 연결을 시도할 대 대기하는 최대 시간</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 시스템 환경에 맞춰 적절한 타임아웃 시간이 설정된 상태</td></tr>
<tr><td>취약
■ 너무 짧거나 긴 타임아웃 설정으로 리소스 낭비 또는 지연 발생할 수 있는 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ 네트워크 지연 및 안정성을 고려하여 설정
■ 네트워크 지연이 큰 환경에서는 타임아웃 증가, 빠른 응답이 필요한 경우 적절히 단축
■ 충분한 테스트를 통해 연결 성능 및 안정성을 검증

설정 예시
■ kafka/config/consumer.properties 와 kafka/config/consumer.properties 에서 아래 예시와 같이 작성
<pre>socket.connection.setup.timeout.max.ms=40000 # 40초
socket.connection.setup.timeout.ms=15000 # 15초</pre></td></tr>
</table>

No 1.7	클라이언트 DNS 조회
개요	
점검 내용	■ 클라이언트가 브로커의 호스트 이름을 확인할 때 사용할 DNS 조회 모드 결정
보안위협	■ 악의적인 DNS 서버가 DNS 캐시를 변조하여 잘못된 IP 로 연결되어 DNS 스푸핑 발생 가능성 존재 ■ 잘못된 DNS 설정은 연결 실패를 초래하여 브로커와의 연결을 불안정하게 만들 가능성 존재
점검 대상 및 판단 기준	
점검 대상	■ Producer client, Consumer client, kafka/config/producer.properties, kafka/config/consumer.properties ■ client.dns.lookup(default: use_all_dns_ips) : 브로커 호스트 이름을 확인할 때 DNS 조회 모드 설정 ■ use_all_dns_ips : DNS 조회 시 반환된 모든 IP 주소에 순차적으로 연결을 시도 ■ resolve_cannonical_bootstrap_servers_only : 부트스트랩 서버의 주소를 하나의 정식 이름으로 해석
판단 기준	양호 ■ use_all_dns_ips 또는 resolve_cannonical_bootstrap_servers_only 가 네트워크 환경에 적합하게 설정되어 있으며, 보안상 안전하게 설정된 상태 ■ 신뢰할 수 있는 DNS 서버가 설정되어 있으며, DNS 보안 감사가 주기적으로 이루어지는 상태
	취약 ■ DNS 설정이 부적절하거나 신뢰할 수 없는 DNS 서버를 사용 중
조치 방법	
고려사항 ■ use_all_dns_ips : 네트워크 환경에서 유연성을 필요로 할 때 유용하나, DNS 서버가 변조되었을 경우 악의적인 IP 로의 연결을 시도할 수 있기 때문에 신뢰할 수 있는 DNS 서버를 사용하고 DNS 보안을 강화해야 함 ■ resolve_canonical_bootstrap_servers_only : 부트스트랩 단계에서만 사용되며, 서버 이름을 하나의 정식이름으로 해석하는 방식. 네트워크 안정성을 높일 수 있지만 다중 경로 연결의 유연성을 제한할 수 있음. 해당 값을 설정할 때는 DNS 서버의 신뢰성을 검증하는 것이 중요	

설정 예시

■ kafka/config/consumer.properties 와 kafka/config/producer.properties 에서 아래 예시와 같이 작성

```
client.dns.lookup=use_all_dns_ips
```

No 1.8	메시지 전송 보장
개요	
점검 내용	■ 메시지를 정확하게 한 번만 전송하거나, 중복 없이 순서대로 처리를 보장하는 설정 점검
보안위협	■ 중요 데이터 처리에서 순서 보장이 필요한 시스템에 메시지 순서 불일치 문제 발생 가능 ■ 메시지 중복이 발생할 경우 데이터 무결성에 문제가 발생할 수 있음
점검 대상 및 판단 기준	
점검 대상	■ Producer client, kafka/config/producer.properties ■ enable.idempotence(default: true) : 프로듀서가 메시지를 전송할 때, 동일한 메시지가 중복으로 작성되지 않도록 보장 ■ use_all_dns_ips : DNS 조회 시 반환된 모든 IP 주소에 순차적으로 연결을 시도 ■ max.in.flight.request.per.connection(default: 5) : 클라이언트가 한 연결에서 확인 응답을 받지 않은 요청을 동시에 보낼 수 있는 최대 개수
판단 기준	양호 ■ enable.idempotence 가 true 로 설정되어 있으며, 중복 메시지 발생을 방지하고 데이터의 정확성을 보장한 상태.신뢰할 수 있는 DNS 서버가 설정되어 있으며, DNS 보안 감사가 주기적으로 이루어지는 상태 ■ max.in.flight.request.per.connection 이 5 이하, retries 값이 0 이상, acks 값이 all 로 설정되어있어 충돌 없이 기능이 활성화 된 상태
	취약 ■ enable.idempotence 가 false 로 설정되어있어 메시지가 중복으로 기록될 수 있으며, 이로 인해 데이터 일관성 및 정확성이 저하될 위험이 존재하는 상태 ■ max.in.flight.request.per.connection 가 5 를 초과하고 acks 가 all 이 아닌 경우 등, 충돌하는 설정으로 인해 enable.idempotence 기능이 비활성화 된 상태
조치 방법	
고려사항 ■ enable.idempotence 활성화 ■ max.in.flight.request.per.connection 값을 5 이하로 설정 ■ retries 값을 0 이상으로 설정	

- acks 값을 all 로 설정
- 위 설정을 적용하여 enable.idempotence 가 안전하게 활성화 되도록 점검
- 설정 후 충분한 테스트를 통해 연결 성능 및 안정성을 검증

설정 예시

- kafka/config/consumer.properties 와 kafka/config/producer.properties 에서 아래 예시와 같이 작성

```
enable.idempotence=true
max.in.flight.requests.per.connection=5
retries=3
acks=all
```

3. Broker

3.1. 개요

3 장에서는 Kafka 의 가장 핵심 구성 요소인 브로커에 대해 다룬다. 브로커는 Producer 와 Consumer 간의 데이터 송수신을 중개하고, 클러스터 내에서 데이터를 저장하고 관리하는 역할을 수행한다. 브로커는 Kafka 클러스터의 안정성과 성능을 결정하는 중요한 요소이므로, 통신 프로토콜 관리, 메타데이터 등을 포함한 다양한 측면에서의 관리가 필요하다.

브로커가 제대로 동작하지 않을 경우, 데이터 처리 지연, 메시지 손실, 시스템 장애 등의 문제가 발생할 수 있으며 이는 클라이언트와 전체 Kafka 시스템에 큰 영향을 미친다. 따라서 이 장에서는 Kafka 브로커의 보안과 안정성을 유지하기 위한 핵심 진단 항목을 소개한다.

3.2. Broker 진단 List

분류	소분류	점검 항목
Broker	2.1	통신 프로토콜
	2.2	소켓 통신
	2.3	메타데이터 관리
	2.4	연결 복구
	2.5	설정 변경 정책
	2.6	브로커 허용 최대 메시지 크기
	2.7	브로커 간 하트비트

3.3. Broker 진단 항목

<table>
<tr><th>No 2.1</th><th>통신 프로토콜</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 브로커 간 통신에서 사용할 프로토콜의 안정성과 최신 버전 사용 점검</td></tr>
<tr><td>보안위협</td><td>■ PLAINTEXT 사용 시 네트워크 상에서 데이터가 평문으로 노출될 가능성 존재</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>Consumer client, Broker, kafka/config/consumer.properties, kafka/config/kraft/server.properties
■ security.inter.broker.protocol(default: null) : 브로커 간 통신에서 사용되는 보안 프로토콜
■ inter.broker.protocol.version(default: 3.8-IV0A) : 브로커 간 통신에서 사용한 보안 프로토콜 버전</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ security.inter.broker.protocol 값이 PLAINTEXT 이외의 값으로 설정되어 있는 상태
■ inter.broker.protocol.version 값이 가장 최신 버전이거나 취약점이 발견되지 않은 버전으로 설정된 상태</td></tr>
<tr><td>취약
■ security.inter.broker.protocol 값이 PLAINTEXT 로 설정되어 통신 과정에 어떠한 암호화도 걸리지 않은 상태
■ inter.broker.protocol.version 값이 취약점이 발견된 버전으로 설정된 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">security.inter.broker.protocol 변경
■ security.inter.broker.protocol 은 기본적으로 PLAINTEXT 로 설정됨
■ 해당 값을 데이터에 따라 알맞은 보안 통신을 선택하여 설정
inter.broker.protocol.version 설정
■ 클러스터 내 모든 브로커는 동일한 값을 가져야 함
■ 브로커 간 통신에서 호환성 문제가 발생하지 않도록 현재 사용하는 프로토콜 버전에 유의하여 업데이트 해야만 함</td></tr>
</table>

No 2.2	소켓 통신
개요	
점검 내용	■ 소켓 통신 관련 설정을 확인하여 송/수신 버퍼 크기와 클라이언트 요청 크기가 적절히 설정되어 있는지 점검
보안위협	■ 송수신 버퍼 크기 설정이 부적절한 경우, 네트워크 병목 현상이 발생하거나 데이터가 손실될 가능성 존재 ■ 버퍼 크기나 요청 크기가 실제 트래픽에 비해 과도하게 설정되면 메모리 리소스가 낭비될 가능성 존재
점검 대상 및 판단 기준	
점검 대상	■ Broker, kafka/config/kraft/server.properties ■ socket.request.max.bytes(default: 104857600) : 클라이언트가 브로커에 전송할 수 있는 요청의 최대 크기 ■ socket.send.buffer.bytes(default: 102400) : 소켓 서버가 사용하는 송신 버퍼의 크기 ■ socket.receive.buffer.bytes(default: 102400) : 소켓 서버가 사용하는 수신 버퍼의 크기
판단 기준	양호 ■ 소켓 송수신 버퍼의 값이 네트워크 트래픽 및 대역폭에 적합한 크기로 설정되어있는 상태 ■ 최대 요청 크기가 java heap 보다 작고 message.max.bytes 값과 크거나 같은 경우
	취약 ■ 송수신 버퍼 크기가 지나치게 작아 네트워크 병목 현상을 발생시키거나 필요 이상으로 크게 설정하여 메모리를 낭비하는 상태 ■ 최대요청 크기가 java heap 보다 크거나 message.max.bytes 보다 작은 경우
조치 방법	
고려 사항 ■ 시스템의 네트워크 환경과 트래픽 패턴에 맞춰 테스트를 통해 송수신 버퍼 적정 크기값을 도출 후 설정 ■ 클러스터에서 사용중인 메시지 크기보다 크거나 같은 값으로 최대 요청 크기 설정 설정 예시 ■ kafka/config/kraft/server.properties 에 아래 예시와 같이 작성	

```
socket.request.max.bytes=52428800  # 50MB로 설정 (메시지 크기에 맞춤)
socket.send.buffer.bytes=131072  # 128KB로 설정 (네트워크 환경 고려)
socket.receive.buffer.bytes=131072  # 128KB로 설정 (네트워크 환경 고려)
```

No 2.3	메타데이터 관리
개요	
점검 내용	■ 중요 정보를 담고 있는 메타데이터의 생존 시간, 유휴 시간 복구에 관한 설정이 적절한지 점검
보안위협	■ 오래된 메타데이터는 네트워크 지연의 원인이 될 수 있을 뿐만 아니라 동시성 문제의 원인이 될 수 있음 ■ 너무 자주 메타데이터를 갱신할 경우 네트워크의 부담 가중되어 많은 리소스 자원을 필요로 함
점검 대상 및 판단 기준	
점검 대상	■ Broker, kafka/config/kraft/server.properties ■ metadata.max.age.ms(default: 300000) : 메타데이터의 최대 생존 시간 ■ metadata.max.idle.ms(default: 300000) : 메타데이터가 가장 최근 업데이트 된 후 최대 유휴 시간 ■ metadata.recovery.strategy(default: none) : 모든 브로커가 사용 불가능한 경우 메타데이터 복구 전략 제어 설정
판단 기준	양호 ■ 메타데이터 생존 시간과 유휴 시간이 적절하게 설정된 상태 ■ 환경에 맞는 메타데이터 복구 전략으로 설정된 상태
	취약 ■ 메타데이터 생존 시간과 유휴 시간이 지나치게 짧거나 긴 상태 ■ 환경에 맞는 메타데이터 복구 전략이 제대로 설정되지 않은 상태
조치 방법	
metadata.max.age.ms 설정 ■ 최신 메타데이터를 유지해 보안 정책을 준수해야 함 metadata.max.idle.ms ■ 클러스터 환경에서 메타데이터 업데이트가 오래되지 않도록 적절한 주기로 설정 적절한 metadata.recovery.strategy 설정 ■ NONE : 클라이언트 복구를 시도하지 않기 때문에 브로커의 변경 가능성이 적고 실패 시 빠르게 문제를 탐지해야 하는 환경에서 사용 ■ REBOOTSTRAP : 부트스트랩 프로세스로 새 브로커로 연결을 시도. 브로커의 변경이 빈번하거나 클라이언트가 오랜 시간 비활성 상태로 있다가 다시 활성화될 가능성이 높은 연결의 경우 사용	

<table>
<tr><th>No 2.4</th><th>연결 복구</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 네트워크 연결이 끊겼을 경우 브로커와 클라이언트 간의 통신 복구 설정이 제대로 이루어지는 지 점검</td></tr>
<tr><td>보안위협</td><td>■ 지속적인 연결 실패할 경우 네트워크 로그와 브로커 IP 포트를 드러내 공격자 악용 가능성 존재</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Broker, kafka/config/kraft/server.properties
■ reconnect.backoff.max.ms(default: 1000) : 호스트에 재연결 시 기다리는 최대 시간
■ reconnect.backoff.ms(default: 50) : 호스트에 재연결 시 기다리는 기본 시간
■ retry.backoff.max.ms(default: 1000) : 호스트에 재요청 시 기다리는 최대 시간
■ retry.backoff.ms(default: 100) : 호스트에 재요청 시 기다리는 기본 시간</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 반복적인 연결/요청 시도가 브로커를 과부하 상태로 몰고 가지 않도록 최대값을 제한한 상태
■ 적절한 초기 값을 설정하여 자원을 적당하게 사용하는 상태</td></tr>
<tr><td>취약
■ 적당히 높지 않거나 너무 높은 부적절한 최대값으로 브로커가 과부하 상태로 이어질 수 있는 상태
■ 너무 짧은 간격으로 연결/요청 시도가 이루어져 자원 남용으로 이어진 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ 재연결 대기 시간이 짧을 경우 브로커 자원 낭비, DoS 발생 가능성이 존재.
■ 길 경우 클라이언트의 재연결 지연으로 서비스 가용성 저하의 가능성
■ 재요청 대기 시간이 짧을 경우 요청 폭주로 인해 브로커 처리 성능 저하 가능
■ 재요청 대기 시간이 길 경우 클라이언트의 요청 지연으로 서비스 가용성 저하의 가능성

적절한 설정값
■ 테스트 환경에서 보안/안정성 균형을 고려하여 클러스터 트래픽, 인증 방식, 네트워크 안정성에 따라 세부 조정을 진행하는 걸 추천
■ 연결 시도 실패 빈도 및 재요청 트래픽 패턴을 분석하여 최적화 진행 추천</td></tr>
</table>

설정 예시

- kafka/config/kraft/server.properties 에 아래 예시와 같이 작성

```
# 재연결 대기 시간 설정 (최대 5초, 기본 200ms)
reconnect.backoff.max.ms=5000
reconnect.backoff.ms=200

# 재요청 대기 시간 설정 (최대 2초, 기본 300ms)
retry.backoff.max.ms=2000
retry.backoff.ms=300
```

No 2.5	설정 변경 정책
개요	
점검 내용	■ 리소스의 설정을 변경할 때, alter.config.policy.class.name 설정을 통해 정의된 정책이 적용되고 있는지 점검
보안위협	■ 부적절한 설정 변경으로 인해 리소스를 낭비하거나 시스템 성능을 저하시킬 가능성 존재 ■ 중요한 보안 설정이 무분별하게 변경될 경우, 클러스터의 보안이 악화될 가능성 존재
점검 대상 및 판단 기준	
점검 대상	■ Broker, kafka/config/kraft/server.properties ■ alter.config.policy.class.name(default: null) : 리소스의 설정 변경을 제어하기 위한 정책 클래스를 지정
판단 기준	양호 ■ alter.config.policy.class.name 설정이 활성화 되어있고, 정책 클래스가 적절히 작동하는 상태
	취약 ■ 설정이 비활성화되어 있거나, 정책 클래스가 정의되지 않았거나 잘못 구현됨
조치 방법	

고려 사항

■ 설정 변경 제어를 위해 적절한 정책 마련

설정 예시

Step 1) 아래 예시와 같이 AlterConfigPolicy 인터페이스 구현

■ 아래 예시는 데이터 보존 시간(retention.ms)가 1 시간 이상이어야 한다는 정책 클래스 구현

```
import org.apache.kafka.common.errors.PolicyViolationException;
import org.apache.kafka.server.policy.AlterConfigPolicy;
import java.util.Map;
public class RetentionPolicy implements CustomAlterConfigPolicy {
   @Override
   public void validate(RequestMetadata requestMetadata) throws PolicyViolationException {
      String retentionMs = requestMetadata.configs().get("retention.ms");
      if (retentionMs != null && Long.parseLong(retentionMs) < 3600000) {  // 1 시간 =
3600000ms
```

```
            throw new PolicyViolationException("Retention time must be at least 1 hour.");
        }
    }
    @Override
    public void configure(Map<String, ?> configs) {
        // 추가 설정 필요 시 구현
    }
    @Override
    public void close() {
        // 리소스 정리 필요 시 구현
    }
}
```

Step 2) kafka/config/kraft/server.properties 에 정책 클래스 설정

■ 아래 예시와 같이 설정

```
alter.config.policy.class.name=com.example.kafka.policy.CustomRetentionPolicy
```

<table>
<tr><th>No 2.6</th><th>브로커 허용 최대 메시지 크기</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ Kafka 브로커에서 처리할 수 있는 메시지의 최대 크기 설정</td></tr>
<tr><td>보안위협</td><td>■ 지나치게 큰 메시지는 브로커와 클라이언트 사이의 과부하 및 서비스 거부 유발 가능
■ 지나치게 작은 메시지는 네트워크와 시스템 리소스를 비효율적으로 사용할 가능성 존재</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Broker, kafka/config/kraft/server.properties
■ message.max.bytes(default: 104858) : 브로커가 허용하는 최대 메시지 배치 크기
■ 해당 크기는 압축이 활성화된 경우 압축 후의 크기가 기준</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ message.max.bytes 값이 시스템 환경에 적합하게 설정되어 있으며, 크기 및 배치 크기가 효율적으로 관리되고 있는 상태</td></tr>
<tr><td>취약
■ 설정값이 너무 크거나 작게 설정되어 있어 성능 저하 또는 과도한 리소스를 소모하는 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ 시스템 환경에 적합하게 메시지 크기를 적절히 제한
■ 클러스터에서 처리할 수 있는 메시지 크기가 애플리케이션 요구 사항을 충족하는지 확인

설정 예시
■ kafka/config/kraft/server.properties 에 아래 예시와 같이 작성
<pre>message.max.bytes=2097152 # 2MB</pre></td></tr>
</table>

<table>
<tr><th>No 2.7</th><th>브로커 간 하트비트</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 브로커 간 하트비트 주기를 설정하는 항목 점검</td></tr>
<tr><td>보안위협</td><td>■ 지나치게 짧은 하트비트 간격은 브로커 간 불필요한 네트워크 트래픽 증가 및 시스템 부하로 인한 장애 발생 가능성 증가
■ 지나치게 긴 하트비트 간격은 브로커 간 연결 상태를 늦게 감지하여 서비스 장애가 발생할 위험 존재</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Broker, kafka/config/kraft/server.properties
■ broker.heartbeat.interval.ms(default: 2000) : 브로커 간의 하트비트 간격을 설정</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 설정 값이 네트워크 환경에 적절히 설정되어 있으며 하트비트 간격이 너무 짧거나 길지 않아 브로커의 안정성을 확보한 상태</td></tr>
<tr><td>취약
■ 설정 값이 지나치게 짧거나 길게 설정되어서 네트워크 부하 또는 서비스 장애 감지가 늦은 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ 네트워크 환경을 고려하여 적절히 설정
■ 레이턴시가 높은 환경에서는 값을 늘려야 안정적인 연결 유지 가능
■ 클러스터에서 브로커의 수가 많을 경우 값을 적절히 늘려 네트워크 부하를 줄이도록 고려

설정 예시
■ kafka/config/kraft/server.properties 에 아래 예시와 같이 작성
<pre>broker.heartbeat.interval.ms=3000 #(3초)</pre></td></tr>
</table>

4. Topic

4.1. 개요

4 장에서는 Kafka 의 데이터 흐름을 정의하는 기본 단위인 Topic 에 대해 다룬다. 토픽은 Kafka 에서 데이터가 저장되고 교환되는 논리적 단위로, Producer 가 메시지를 전송하고 Consumer 가 메시지를 소비하는 지점이다. 적절한 토픽 관리 없이는 메시지 손실, 데이터 정합성 문제, 성능 저하 등의 위험이 발생할 수 있다.

Kafka 토픽은 메시지 스트림을 효율적으로 분배하고 관리하는 데 중요한 역할을 하며, 특히 파티션, 복제, 스토리지 설정은 토픽의 성능과 보안에 큰 영향을 미친다. 이 장에서는 Kafka 토픽을 안전하게 보호하고 효과적으로 운영하기 위한 보안 관리 방안과 진단 항목을 소개한다.

4.2. Topic 진단 List

분류	소분류	점검 항목
Topic	3.1	내부 토픽 접근 관리
	3.2	토픽 자동 생성
	3.3	레코드 배치 크기 (Topic)
	3.4	토픽 생성 정책

4.3. Topic 진단 항목

<table>
<tr><th>No 3.1</th><th>내부 토픽 접근 관리</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 컨슈머가 내부 토픽에(예: __consumer_offsets)을 읽을 수 있는지 여부를 제어</td></tr>
<tr><td>보안위협</td><td>■ 내부 토픽에는 클러스터 상태나 오프셋 정보와 같은 민감한 메타데이터가 포함됨
■ 민감한 정보가 외부로 유출되지 않게 관리</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Consumer client, kafka/config/consumer.properties
■ exclude.internal.topics(default: true) : Consumer 가 내부 토픽을 읽을 수 있는지 여부를 설정</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ exclude.internal.topics = true 로 설정되어 있으며, consumer 가 내부 토픽에 접근할 수 없는 상태</td></tr>
<tr><td>취약
■ exclude.internal.topics = false 로 설정되어 있거나, 내부 토픽에 대한 접근이 허용된 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">exclude.internal.topics 활성화
■ kafka/config/consumer.properties 에서 exclude.internal.topics=true 추가
<pre>exclude.internal.topics=true # 내부 토픽 비활성화</pre>
ACL 설정
■ 내부 토픽에 대한 클라이언트의 접근을 명확히 제한하도록 ACL 설정</td></tr>
</table>

<table>
<tr><th>No 3.2</th><th>토픽 자동 생성</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 클라이언트가 브로커에 존재하지 않는 토픽을 구독하거나 할당할 때, 브로커가 해당 토픽을 자동으로 생성할지 여부를 결정</td></tr>
<tr><td>보안위협</td><td>■ allow.auto.create.topics & auto.create.topics.enable 가 true 로 설정된 경우 악의적인 사용자가 불필요한 토픽을 생성 가능
■ 자동으로 생성된 불필요한 토픽으로 인해 클러스터 리소스를 낭비하고 성능에 영향을 줄 가능성 존재</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Consumer client, Broker, kafka/config/consumer.properties, kafka/config/kraft/server.properties
■ allow.auto.create.topic(default: true) : consumer 가 브로커에 존재하지 않는 토픽을 구독할 때, 해당 토픽을 자동으로 생성할지 여부 설정
■ auto.create.topics.enable(default: true) : 클라이언트가 브로커에 존재하지 않는 토픽을 구독하거나 할당하려 할 때, 자동으로 해당 토픽을 생성할 수 있게 설정</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ allow.auto.create.topics 와 auto.create.topics.enable 이 false 로 설정되어 있으며 클라이언트가 불필요한 토픽 생성을 통해 리소스를 낭비하지 않는 상태</td></tr>
<tr><td>취약
■ allow.auto.create.topics 와 auto.create.topics.enable 이 true 로 설정되어 있으며 클라이언트가 불필요한 토픽 생성을 통해 리소스를 낭비하거나 보안 관리의 어려움을 초래할 가능성이 있는 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">allow.auto.create.topics, auto.create.topics.enable 설정
■ 해당 옵션들은 기본적으로 true 로 설정됨
■ 기본적으로는 비활성화하는 것이 안전하나, 개발 및 테스트 환경이나 토픽이 빠르게 처리되어야 하는 환경에 맞게 고려하여 해당 옵션 활성화 여부 결정
■ kafka/config/consumer.properties 에서 allow.auto.create.topics 설정
■ kafka/config/kraft/server.properties 에서 auto.create.topics.enable 설정

설정 예시</td></tr>
</table>

- kafka/config/consumer.properties 에 아래 예시와 같이 작성

```
# 자동 토픽 생성 비활성화 설정
allow.auto.create.topics=false
```

- kafka/config/kraft/server.properties 에 아래 예시와 같이 작성

```
# 자동 토픽 생성 비활성화 설정
auto.create.topics.enable=false
```

<table>
<tr><th>No 3.3</th><th>레코드 배치 크기 (Topic)</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ Kafka에서 허용하는 최대 레코드 배치 크기 설정 점검</td></tr>
<tr><td>보안위협</td><td>■ 지나치게 큰 메시지는 브로커와 클라이언트 사이의 과부하 및 서비스 거부 유발 가능
■ 지나치게 작은 메시지는 네트워크와 시스템 리소스를 비효율적으로 사용할 가능성 존재</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Topic
■ max.message.bytes(default : 1048588) : Kafka 에서 허용하는 최대 레코드 배치 크기
■ 해당 크기는 압축이 활성화 된 경우 압축 후의 크기가 기준</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ max.message.bytes 값이 시스템 환경에 적합하게 설정되어 있으며, 메시지 크기 및 배치 크기가 효율적으로 관리되고 있는 상태</td></tr>
<tr><td>취약
■ 설정값이 너무 크거나 작게 설정되어 있어, 성능 저하 또는 과도한 리소스를 소모하는 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ 시스템 환경에 적합하게 메시지 크기를 적절히 제한
■ max.message.bytes(Topic) 값과 message.max.bytes(Broker) 값을 동일하게 설정하여 데이터 전송 중 오류가 발생하지 않도록 방지

설정 예시
■ 아래 명령어 예시와 같이 설정 가능
<code>bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type topics --entity-name my-topic --alter --add-config max.message.bytes=128000</code></td></tr>
</table>

No 3.4	토픽 생성 정책
개요	
점검 내용	■ 토픽을 생성할 때, create.topic.policy.class.name 설정을 통해 정의된 정책이 적용되고 있는지 점검
보안위협	■ 적절한 토픽 생성 정책이 없으면, 악의적인 사용자가 부적절한 토픽을 생성할 수 있으며, 이로 인해 리스 낭비, 성능 저하 등의 사고 유발 가능 ■ 토픽 생성 정책이 없어 복제본 수가 너무 적은 토픽이 생성될 경우 데이터 손실 위험 존재
점검 대상 및 판단 기준	
점검 대상	■ Broker, kafka/config/kraft/server.properties ■ create.topic.policy.name(default: null) : 토픽 생성 요청에 대한 정책을 정의하는 클래스 이름
판단 기준	양호 ■ create.topic.policy.name 설정이 활성화 되어있고, 정책 클래스가 적절히 작동하는 상태
	취약 ■ 설정이 비활성화 되어있거나, 정책 클래스가 정의되지 않았거나 잘못 구현된 상태
조치 방법	

고려 사항

■ 토픽 이름 및 복제 수, 파티션 수에 대해 설정

설정 예시

Step 1) 아래 예시와 같이 CreateTopicPolicy 인터페이스 구현

■ 아래 예시는 최소 파티션 수를 검증하는 인터페이스

```
import org.apache.kafka.server.policy.CreateTopicPolicy;
import org.apache.kafka.common.errors.PolicyViolationException;
import java.util.Map;
public class CustomCreateTopicPolicy implements CreateTopicPolicy {
   @Override
   public void validate(RequestMetadata requestMetadata) throws PolicyViolationException {
      // 예시: 토픽의 파티션 수가 최소 3 이어야 한다는 정책
      if (requestMetadata.numPartitions() < 3) {
```

```
            throw new PolicyViolationException("Topic must have at least 3 partitions.");
        }
        // 추가적인 검증 로직을 구현할 수 있습니다.
    }
    @Override
    public void configure(Map<String, ?> configs) {
        // 필요 시, 정책 클래스에서 사용할 설정을 처리하는 로직을 구현
    }
    @Override
    public void close() {
        // 정책을 종료할 때 자원 해제 로직을 구현
    }
}
```

Step 2) kafka/config/kraft/server.properties 에 정책 클래스 설정

■ 아래 예시와 같이 설정

```
create.topic.policy.class.name=com.example.kafka.policy.CustomCreateTopicPolicy
```

5. ACL (Access Control List)

5.1. 개요

5 장에서는 kafka 의 보안 모델에서 "인가"를 담당하는 ACL(Access List Control)에 대해 다룬다. ACL 은 Kafka 리소스(예: 토픽, 컨슈머 그룹 등)에 대해 사용자 또는 애플리케이션 권한을 정의하는 방식으로, 데이터 접근 제어를 수행한다. 이를 통해 Kafka 클러스터의 데이터 기밀성과 무결성을 유지할 수 있다.

적절한 ACL 설정은 권한 없는 사용자의 접근을 방지하고, 의도치 않은 데이터 변경이나 삭제를 예방하는 데 매우 중요하다. 특히, 잘못된 ACL 구성은 데이터 노출, 권한 우회와 같은 심각한 보안 문제를 초래할 수 있다. 이 장에서는 Kafka ACL 을 안전하게 적용 및 관리하기 위한 진단 항목을 제공한다.

5.2. 요청 별 최소 필요 권한

Kafka에서 ACL을 적용할 때 최소 권한 원칙을 따르는 것은 보안을 강화하고, 의도치 않은 데이터 접근 및 변경을 방지하는 데 필수적이다. 각 요청이나 행위에 대해 최소한으로 필요한 권한만을 부여함으로써, 권한 과잉 부여로 인한 보안 리스크를 줄이고, Kafka 클러스터의 안정성을 유지할 수 있다.

<부록 1>은 kafka의 각 요청에 대해 필요한 최소 권한을 정리한 List이다. ACL을 설계하거나 관리할 때 해당 리스트를 참고하여 안전하게 설정하는 것을 권장한다.

5.3. ACL 진단 List

분류	소분류	점검 항목
Broker	1.13	최소 권한 원칙
	1.14	Wildcard
	1.15	Super User
	1.16	Cluster 리소스 관리

5.4. ACL 진단 항목

<table>
<tr><th>No 4.1</th><th>최소 권한 원칙</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ ACL 설정을 검토하여 사용자 및 애플리케이션에 부여된 권한이 최소 권한 원칙에 부합하는지 확인</td></tr>
<tr><td>보안위협</td><td>■ 불필요한 권한 부여로 인해 악의적인 사용자나 내부 오류로 데이터 유출, 삭제, 변조 등의 피해가 발생할 수 있음.</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Kafka 에서 ACL 을 통해 관리되는 사용자 및 애플리케이션 권한</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 사용자 및 애플리케이션에 필요한 최소 권한만이 부여되어 있음
■ 위 <표 1. 최소 권한 관리 기준>에 부합하게 설정되어 있음</td></tr>
<tr><td>취약
■ 사용자 또는 애플리케이션에 불필요한 권한이 부여되어 있는 경우
■ 위 <표 1. 최소 권한 관리 기준>에 부합하지 않게 설정되어 있는 경우</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">ACL 을 확인, 부여, 삭제, 수정 등을 위해서 kafka/bin/kafka-acl.sh 스크립트를 사용한다.

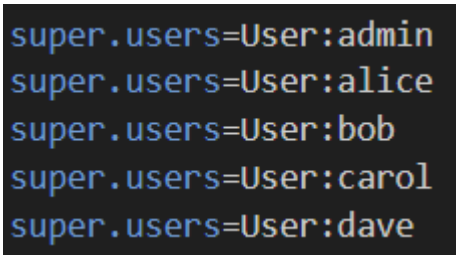
step 1) ACL 확인
■ 시스템에 등록되어 있는 모든 ACL 을 아래의 명령어로 확인 가능하다.
kafka-acls.sh --bootstrap-server <broker-url> --list

step 2) ACL 수정
■ <부록 1. 요청 별 필요 최소 권한>에 부합하지 않게 설정되어 있는 유저 확인.
■ 부합하지 않는 경우 아래의 명령어로 ACL 을 제거하거나 수정한다.
ACL 삭제

kafka-acls.sh --bootstrap-server <broker-url> --remove --allow-principal User:<username> --operation <operation> --topic <topic-name> --group <group-name></td></tr>
</table>

<table>
<tr><th>No 4.2</th><th>Wildcard</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ ACL 설정에서 와일드카드 또는 패턴 매칭이 사용된 권한이 적절하게 설정되어 있는지 확인</td></tr>
<tr><td>보안위협</td><td>■ 와일드카드를 과도하게 사용할 경우 의도하지 않은 리소스에 접근이 허용될 수 있으며, 데이터 유출, 삭제 또는 변조로 이어질 위험이 있음.
■ 불필요한 권한이 부여되면서 리소스 접근이 과도하게 확장될 수 있음</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ ACL 에서 리소스 패턴이 prefixed 또는 와일드카드로 설정된 항목</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 와일드카드 및 패턴 매칭이 필요한 리소스에만 사용되었으며, 의도치 않은 리소스 접근이 발생하지 않도록 제한된 상태
■ 리소스 패턴이 prefixed 로 설정되었을 경우, 접두사 또는 접미사가 명확하고 제한적으로 설정된 상태</td></tr>
<tr><td>취약
■ 와일드카드 및 패턴 매칭이 필요 이상의 권한에 사용되어 불필요한 리소스 접근이 허용된 경우
■ ACL 설정에 명확한 제한이 없어 다른 리소스에 비의도적으로 권한이 부여된 경우</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">권한 최소화
■ 와일드카드 및 패턴 매칭 사용을 최소화 하고, 특정 리소스에 권한을 명시적으로 부여
■ 예: 와일드카드 대신 특정 토픽 리소스를 지정하여 권한 설정
<code>kafka-acls.sh --add --allow-principal User:producer1 --operation WRITE --topic my-topic</code>

제한적 패턴 매칭
■ 리소스 패턴을 prefixed 로 설정 시, 접두사를 신중히 설계하여 제한적으로 사용
<code>kafka-acls.sh --add --allow-principal User:app-user --operation READ --resource-pattern-type prefixed --topic my-app</code></td></tr>
</table>

<table>
<tr><th>No 4.3</th><th>Super User</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ super.users 설정에 정의된 사용자가 과도하게 많은지 확인하고, 필요한 최소 사용자만 슈퍼유저 권한을 가지도록 설정되어 있는지 점검</td></tr>
<tr><td>보안위협</td><td>■ 슈퍼유저 권한이 불필요하게 많은 사용자에게 부여되면, 악의적인 공격이나 클러스터의 모든 리소스에 접근이 가능해져 데이터 유출, 변조, 삭제 등의 심각한 보안 위협 발생 가능</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ super.users 설정에 정의된 사용자 목록</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ super.users 에 정의된 사용자가 최소화 되어있으며, 클러스터 관리에 필수적인 사용자만 슈퍼유저 권한을 보유하고 있는 상태
■ 슈퍼유저 권한을 가진 사용자는 정기적으로 점검하고 필요한 경우 권한 수정</td></tr>
<tr><td>취약
■ super.user 에 과도한 수의 사용자가 정의되어 있어 불필요한 권한이 부여된 경우</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">super.users 검토
■ kafka/config/kraft/server.properties 파일에서 super.users 설정을 확인하고 불필요한 사용자나 권한을 가진 항목을 제거
■ 아래와 예시와 같이 kafka/config/kraft/server.properties 에서 super.users 항목을 확인
 
super.users=User:admin
super.users=User:alice
super.users=User:bob
super.users=User:carol
super.users=User:dave

ACL 을 통한 권한 세분화
■ 슈퍼 유저 목록을 최소화 하고 최대한 ACL 을 통해 권한을 세분화 하여 필요 권한만 부여할 수 있도록 설정</td></tr>
</table>

<table>
<tr><th>No 4.4</th><th>Cluster 리소스 관리</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 사용자에게 부여된 cluster 리소스 권한을 점검</td></tr>
<tr><td>보안위협</td><td>■ 불필요한 사용자에게 클러스터에 대한 권한을 부여하면, 시스템 설정을 변경하거나 리소스를 과도하게 생성하는 등의 악의적인 행동 가능</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ cluster 리소스에 대한 권한을 가진 사용자 및 애플리케이션</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ cluster 리소스에 대한 권한을 필요로 하는 최소한의 사용자에게만 부여되어 있으며, 불필요한 권한이 부여되지 않은 상태</td></tr>
<tr><td>취약
■ 과도하게 많은 사용자에게 클러스터 리소스 권한이 부여되어 시스템 설정 변경이나 리소스 생성 남용이 발생할 위험이 있는 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">권한 최소화
■ 클러스터 리소스에 대한 권한을 최소화하여, 필요한 사용자에게만 권한을 부여

권한 검토 및 정기적 검사
■ 정기적으로 Cluster 리소스 권한을 검토하여 불필요한 권한이 부여되지 않도록 관리
kafka-acl.sh --list --resource-cluster
■</td></tr>
</table>

No 4.5	Describe 권한 관리
개요	
점검 내용	■ Describe 권한은 리소스의 메타데이터를 읽을 수 있는 권한 제공, 주로 리소스의 상태나 속성 정보를 조회하는 데 사용 ■ 해당 권한은 READ, WRITE, DELETE, ALTER 권한을 부여할 경우 자동으로 부여됨(ALTER_CONFIG를 부여할 경우, DESCRIBE_CONFIGS가 부여됨)
보안위협	■ 해당 권한이 부여된 사용자는 리소스의 메타데이터를 확인할 수 있으며, 해당 정보가 노출될 시 민감한 데이터나 시스템 구조에 대한 정보 유출 발생 가능 ■ 해당 권한이 남용될 경우 리소스 상태에 대한 세부 사항 파악을 통해 서비스의 취약점을 찾아내거나 공격을 위한 정보를 수집할 수 있음
점검 대상 및 판단 기준	
점검 대상	■ READ, WRITE, DELETE, ALTER, ALTER_CONFIG 권한이 부여된 사용자
판단 기준	양호 ■ 민감한 리소스에 대해 Describe 권한이 불필요하게 부여되지 않으며, 권한이 신중하게 관리된 상태
	취약 ■ READ, WRITE 권한 부여시 자동으로 DESCRIBE 권한이 부여되어, 민감한 리소스에 대해 과도한 정보 접근이 가능해진 상태
조치 방법	
고려 사항 ■ Describe 권한이 불필요한 사용자에게 부여되지 않도록 권한을 최소화 ■ READ, WRITE, DELETE, ALTER, ALTER_CONFIG 부여 시, DESCRIBE 권한이 함께 부여된다는 것을 인지하고 권한 설정	

<table>
<tr><th>No 4.6</th><th>불필요한 계정 관리</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 오래 사용되지 않거나 불필요한 계정을 식별하고 삭제했는지 점검</td></tr>
<tr><td>보안위협</td><td>■ 불필요한 계정에 위험한 권한이 부여된 상태로 남아있을 경우, 공격자가 해당 계정을 악용할 가능성 존재
■ 사용되지 않는 계정이 접근 권한을 가지고 있을 경우, 외부 공격자나 악의적인 내부자가 이를 이용하여 시스템에 침입할 가능성 존재</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Kafka 사용자 계정, Broker log</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 불필요한 계정이 비활성화 되었거나 삭제된 경우
■ 최소한의 사용자만 필요한 권한을 가진 상태
■ 주기적으로 계정 검토가 이루어지는 경우</td></tr>
<tr><td>취약
■ 오랫동안 활동이 없는 계정이나 필요 없는 계정이 여전히 활성화 된 경우
■ 주기적인 계정 검토 및 정리가 이루어지지 않는 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">로그 분석
■ kafka log 에서 사용자의 활동을 추적하고 오래된 계정이나 활동이 없는 계정 식별
<code>grep "User:" /path/to/kafka/logs/server.log</code>
■ 또한, ACL 을 명확히 설정하여 보안을 강화

Consumer Group 의 오프셋 분석
■ kafka-consumer-groups.sh 명령어를 사용하여 특정 Consumer Group 의 마지막 오프셋 정보를 확인
<code>bin/kafka-consumer-groups.sh --bootstrap-server <broker_address> --describe --group <consumer_group_name></code>

주기적인 계정 점검
■ 위 방법 외에도 주기적인 계정 점검 체계 필요</td></tr>
</table>

No 4.7	ACL 미적용 리소스 접근 허용
개요	
점검 내용	■ ACL이 정의되지 않은 리소스에 대해 기본 접근 제어 방식을 결정
보안위협	■ ACL 이 없는 리소스에 모든 사용자가 접근할 경우 민감한 데이터, Topic 등 리소스가 무방비하게 노출 ■ 의도치 않은 데이터 유출 및 시스템 무단 접근 가능성 증가
점검 대상 및 판단 기준	
점검 대상	■ Broker, kafka/config/kraft/server.properties
판단 기준	양호 ■ allow.everyone.if.no.acl.found=false 로 설정되어 있으며, 모든 리소스에 대해 적절한 ACL 이 정의 되어있는 상태 ■ 설정이 false 라도 ACL 이 적절히 정의되어 있지 않은 상태
	취약 ■ allow.everyone.if.no.acl.found=true 로 설정되어 있으며, ACL 이 설정되지 않은 리소스가 존재하는 상태 ■ 설정이 false 라도 ACL 이 적절히 정의되어 있지 않은 상태
조치 방법	
allow.everyone.if.no.acl.found 비활성화 ■ kafka/config/kraft/server.properties 에 아래 코드 추가 kafka/config/kraft/broker.properties = false ■ ACL 을 적절히 설정하여 ACL 이 설정되지 않은 리소스가 없도록 하며, 명확한 접근제어 설정	

6. SSL (Secure Sockets Layer)

6.1. 개요

6 장에서는 Kafka 에 적용할 수 있는 암호화 시스템인 SSL 에 대해 다룬다. SSL 은 네트워크 상의 데이터 전송을 암호화하여 중간자 공격, 데이터 도청, 데이터 변조 등의 보안 위협을 방지하기 위한 프로토콜이다. Kafka 는 데이터 전송의 기밀성과 무결성을 보장하기 위해 SSL 을 적용할 수 있으며, 이를 통해 클라이언트와 브로커, 브로커 간의 통신을 안전하게 보호할 수 있다. 또한 클라이언트와 브로커 간의 상호 인증을 지원하여, 신뢰할 수 없는 사용자나 시스템이 네트워크에 접근하지 못하도록 방지한다. 이를 통해 Kafka 클러스터의 보안을 한층 강화할 수 있다.

이 장에서는 SSL 적용 방법과 설정 절차를 단계별로 설명하며, 주요 진단 항목과 올바른 설정 여부를 확인하기 위한 점검 리스트를 제공한다. 이를 통해 Kafka 사용자들이 SSL 을 효과적으로 구성하고, 안전한 데이터 전송 환경을 구축할 수 있도록 지원한다.

6.2. SSL 적용 가이드라인

아래는 Kafka 에 SSL 을 적용하는 예제이다. 해당 가이드라인은 단순한 예제일 뿐이며, 시스템 환경이나 고려 사항에 맞게 잘 적용하여 안전한 데이터 전송 환경을 구축하는 것을 권장한다.

Step 1) SSL 인증서 생성

Open SSL을 사용하여 SSL 인증서 생성

```
openssl req -new -x509 -keyout ca-key -out ca-cert -days 365
```

- ca-key : 개인 키를 저장할 파일
- ca-cert : 생성된 인증서를 저장할 파일
- 365 : 인증서의 유효 기간

Step 2) 트러스트 스토어 생성

■ 트러스트 스토어는 SSL 통신을 할 때 신뢰할 수 있는 인증서 저장소 역할이며 CA 인증서를 여기에 추가

```
keytool -keystore kafka.broker0.truststore.jks -alias ca-cert -import -file ca-cert
```

- truststore.jks : 트러스트 스토어 파일
- ca-cert : 이전 단계에서 생성한 CA 인증서

Step 3) 키스토어 생성

Kafka 브로커에 사용할 키 저장소 생성. 해당 과정에서 개인 키와 서버 인증서를 포함하는 키스

토어 생성

```
keytool -keystore kafka.broker0.keystore.jks -alias broker -validity 365 -genkey -keyalg RSA -ext SAN=dns:localhost
```

- keystore.jks : 키 저장소 파일
- RSA : 사용할 키 알고리즘
- SAN=dns:localhost : SSL 인증서에 대한 추가 이름

Step 4) CSR(Certificate Signing Request) 생성 및 서명

키스토어에서 CSR을 생성하고, 이를 CA 인증서로 서명하여 브로커의 최종 인증서를 생성

```
keytool -keystore kafka.broker0.keystore.jks -alias broker -certreq -file ca-request-broker
```

CA 인증서로 CSR 서명

```
openssl x509 -req -CA ca-cert -CAkey ca-key -in ca-request-broker -out ca-signed-broker -days 365 -CAcreateserial
```

Step 5) 인증서 키스토어에 추가

서명된 인증서를 키스토어에 추가. Kafka 브로커는 해당 인증서를 사용하여 SSL/TLS 연결 설정 가능

```
keytool -keystore kafka.broker.keystore.jks -alias ca-cert -import -file ca-cert
keytool -keystore kafka.broker.keystore.jks -alias broker -import -file ca-signed-broker
```

Step 6) Broker 설정

■ kafka/config/kraft/server.properties

```
#################### Server Basics #############################
# Kafka node roles (broker, controller)
process.roles=broker,controller
# Node ID for the server instance
node.id=1
# Controller quorum (for KRaft mode)
controller.quorum.voters=1@localhost:9093
############################## SSL & Socket Server Settings
#############################
# SSL truststore and keystore configuration
ssl.truststore.location=/path/to/your/kafka.broker.truststore.jks
ssl.truststore.password=your_truststore_password_here
```

```
ssl.keystore.location=/path/to/your/kafka.broker.keystore.jks
ssl.keystore.password=your_keystore_password_here
ssl.key.password=your_key_password_here
# SSL configuration for client authentication and encryption protocol
ssl.client.auth=required
ssl.enabled.protocol=TLSv1.2
# Define inter-broker listener using SSL
inter.broker.listener.name=SSL
############################## Listeners & Ports
##############################
# Define listeners for broker and controller roles, and their corresponding ports
listeners=SSL://:9092,CONTROLLER://:9093
# Advertised listeners for clients to connect
advertised.listeners=SSL://localhost:9092,CONTROLLER://localhost:9093
# Controller listener names for KRaft mode
controller.listener.names=CONTROLLER
# Listener security protocol mapping
listener.security.protocol.map=INTERNAL:SSL,EXTERNAL:SSL,CONTROLLER:SSL
############################## Log Basics ##############################
# Directory for storing Kafka logs
log.dirs=/tmp/kraft-combined-logs
# Number of partitions and recovery threads
num.partitions=1
num.recovery.threads.per.data.dir=1
############################## Internal Topic Settings
##############################
# Replication factor for internal Kafka topics
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
# Log flush and retention settings
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
```

6.3. SSL 진단 항목 List

분류	소분류	점검 항목
SSL	5.1	클라이언트 인증 요구 설정
	5.2	SSL 프로토콜
	5.3	프로토콜 매핑
	5.4	SSL 인증서 관리

6.4. SSL 진단 항목

No 5.1	클라이언트 인증 요구 설정
개요	
점검 내용	■ 클라이언트의 인증 요구 여부 설정 점검
보안위협	■ 인증을 요구하지 않을 경우, 신뢰할 수 없는 클라이언트가 kafka 클러스터에 접근 가능
점검 대상 및 판단 기준	
점검 대상	■ Broker, kafka/config/kraft/server.properties ■ ssl.client.auth(default: none) : 클라이언트 인증 요구 여부를 설정 ■ required : 클라이언트 인증 필수 ■ requested : 클라이언트 인증 요청 ■ none : 클라이언트 인증 없음
판단 기준	양호 ■ 클라이언트 인증이 필수로 설정되어 있으며, 인증서를 제공해야 Kafka 브로커에 접근 가능
	취약 ■ 클라이언트 인증이 선택적이거나 인증 요구가 비활성화 된 경우
조치 방법	
ssl.client.auth 설정 ■ ssl.client.auth 를 required 로 설정하여 모든 클라이언트가 인증서를 제공해야 브로커에 접근할 수 있도록 설정 ■ kafka/config/kraft/server.properties `ssl.client.auth=required`	

<table>
<tr><th>No 5.2</th><th>SSL 프로토콜</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ SSL 프로토콜 설정 값이 최신 및 권장 프로토콜로 구성되어 있는지 점검
■ 보안 취약점이 있는 이전 프로토콜이 설정 목록에 포함되어 있는지 확인</td></tr>
<tr><td>보안위협</td><td>■ 취약점이 발견된 구 버전 프로토콜은 공격자가 데이터 통신을 도청하거나 조작할 가능성이 높음</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Broker, kafka/config/kraft/server.properties
■ ssl.enabled.protocols(default: TLSv1.2) : 클라이언트 인증 요구 여부를 설정</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ TLSv1.2 또는 TLSv1.3 으로 설정되어있으며 발견된 취약점이 없는 최신 프로토콜 유지</td></tr>
<tr><td>취약
■ 설정이 비어 있거나 불필요한 구버전 프로토콜이 포함된 상태
■ 보안상 안전하지 않은 프로토콜을 사용하는 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ 보안상 안전한 최신 프로토콜 유지
■ Java 8 버전 이하의 경우 TLSv1.2 만 지원 가능
■ Java 11 버전 이상의 경우 TLSv1.2, TLSv1.3 설정 가능
■ kafka/config/kraft/server.properties
ssl.enabled.protocols =TLSv1.3</td></tr>
</table>

<table>
<tr><th>No 5.3</th><th>프로토콜 매핑</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ 리스너별 보안 프로토콜 매핑 적절성 점검</td></tr>
<tr><td>보안위협</td><td>■ 내부 트래픽에 PLAINTEXT 와 같은 낮은 보안 프로토콜이 사용될 경우 민감 데이터 유출 가능성 증가</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Broker, kafka/config/kraft/server.properties
■ listener.security.protocol.map : 리스너 이름과 보안 프로토콜 간의 매핑을 정의</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 내부 트래픽은 INTERNAL:SSL 또는 INTERNAL:SASL_SSL 과 같은 암호화된 프로토콜 설정
■ 외부 트래픽은 EXTERNAL:SASL_SSL 과 같이 인증 및 암호화가 활성화된 프로토콜 사용
■ 각 리스너의 이름이 명확하게 정의되고, 필요한 보안 수준에 맞는 프로토콜에 매핑되어 있는 상태</td></tr>
<tr><td>취약
■ 모든 리스너가 PLAINTEXT 와 같이 암호화 되지 않는 프로토콜로 설정
■ 내부와 외부 트래픽이 동일한 리스너에 동일한 프로토콜로 설정되어 트래픽 분리가 이루어 지지 않은 상태
■ 명시적 매핑 없이 기본값을 그대로 사용하여 환경에 맞지 않는 보안 수준이 적용된 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">고려 사항
■ 기본적으로 암호화된 프로토콜 사용을 권장
■ 내외부 트래픽을 분리하여 네트워크 공격으로부터 클러스터 보호
■ 인증이 필요한 경우 SASL_SSL 프로토콜 사용
■ kafka/config/kraft/server.properties 에 아래 예시와 같이 코드 추가
listener.security.protocol.map=INTERNAL:SSL,EXTERNAL:SASL_SSL,CONTROLLER:SSL</td></tr>
</table>

<table>
<tr><th>No 5.4</th><th>SSL 인증서 관리</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ SSL 인증서의 유효 기간을 주기적으로 점검하고, 만료 전에 적절히 갱신하거나 교체하는지 확인</td></tr>
<tr><td>보안위협</td><td>■ 만료된 인증서를 사용할 경우 클러스터와 클라이언트 간의 안전한 통신이 불가능해짐
■ 비인가 사용자가 통신을 가로채거나 위조된 인증서를 사용하여 중간자 공격 수행 가능</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Broker, kafka/config/kraft/server.properties, SSL 인증서 유효 기간</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 인증서 유효 기간이 만료되지 않았으며, 갱신 주기가 정기적으로 관리되고 있는 상태
■ 인증서 갱신 프로세스가 명확히 수립되어 있는 상태</td></tr>
<tr><td>취약
■ 만료된 인증서를 사용중이거나 인증서의 유효 기간이 임박한 상태
■ 인증서 갱신 절차가 명확하지 않아 만료 후 서비스 중단 가능성이 존재하는 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">인증서 유효 기간 확인
■ 아래 명령어를 통해 인증서의 유효 기간 확인 가능
<code>openssl x509 -in <certificate-file> -noout -dates</code>
■ 실행 예제 사진
 
</td></tr>
</table>

7. SASL (Simple Authentication and Security Layer)

7.1. 개요

7 장에서는 Kafka 에 적용할 수 있는 인증 시스템인 SASL 에 관해 다룰 것이다. SASL 은 네트워크 통신에서 인증을 수행하기 위한 프레임워크로, 다양한 인증 메커니즘을 제공한다. Kafka 는 클러스터 간의 인증과 사용자 인증을 처리를 위해 SASL 적용이 가능하며 이를 통해 시스템의 보안을 강화한다. SASL 은 다양한 인증 메커니즘을 지원하며, 시스템 요구사항에 맞는 유연한 인증 방식을 선택할 수 있도록 한다.

이 장에서는 각 SASL 메커니즘을 소개하고, 설정 방법, 그리고 어떤 환경과 사용자가 어떤 메커니즘을 선택하는 것이 적합한지에 대해 설명할 것이다. 이 장을 통해 Kafka 에 적용 가능한 다양한 SASL 메커니즘 방식을 이해하고, 각 환경에 맞는 최적의 설정을 적용하는 방법을 제시할 것이다

7.2. SASL 설정 가이드라인

아래는 Kafka 에 SASL 을 적용하는 예제이다. 해당 예제는 PLAIN 를 기준으로 설명하고 있으며 시스템 환경이나 고려 사항에 맞게 잘 적용하여 안전한 데이터 전송 환경을 구축하는 것을 권장한다.

Step 1) Broker 설정 추가

■ kafka/config/kraft/server.properties

```
##################### Server Basics #####################################
# Kafka node roles (broker, controller)
process.roles=broker,controller
# Node ID for the server instance
node.id=1
# Controller quorum (for KRaft mode)
controller.quorum.voters=1@localhost:9093
################### Listeners & Ports ###########################
# Define listeners for broker and controller roles, and their corresponding ports
# SASL_PLAIN or SASL_SSL
listeners=SASL_PLAINTEXT://:9092,CONTROLLER://:9093
# Define inter-broker listener using SASL (or using both SASL and SSL)
inter.broker.listener.name=SASL_PLAINTEXT # or SASL_SSL

# Advertised listeners for clients to connect
# SASL_PLAIN or SASL_SSL
advertised.listeners=SASL_PLAINTEXT://192.168.0.68:9092,CONTROLLER://192.168.0.68:9093
# Controller listener names for KRaft mode
controller.listener.names=CONTROLLER
# Listener security protocol mapping
listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:S
ASL_PLAINTEXT,SASL_SSL:SASL_SSL
################### SASL ###########################
# Sets the SASL mechanism for inter-broker communication to PLAIN
sasl.mechanism.inter.broker.protocol=PLAIN
```

```
# Specifies the SASL mechanisms enabled for the Kafka broker (PLAIN is enabled here)
sasl.enabled.mechanisms=PLAIN
```

- sasl.mechanism.inter.broker.protocol : 브로커 내부 통신 간의 protocol 시 사용할 SASL에 대해 정의
- sasl.enabled.mechanisms : SASL을 사용할 경우 사용한 매커니즘

Step 2) Broker kafka_server_jaas.conf 작성

■ user_{username} = "{password}"의 형식으로 사용자를 추가할 수 있다.

■ kafka/config/kafka_server_jaas.conf

```
KafkaServer {
    org.apache.kafka.common.security.plain.PlainLoginModule required
    username="admin"
    password="admin-secret"
    user_admin="admin-secret"
    user_user1="user1-secret";
};
```

Step 3) Client 설정 추가

■ kafka/config/client(producer or consumer).properties

```
security.protocol=SASL_PLAIN # or using both SASL and SSL
sasl.mechanism=PLAIN
```

Step 4) Client kafka_server_jaas.conf 작성

■ kafka/config/kafka_server_jaas.conf

```
KafkaClient {
    org.apache.kafka.common.security.plain.PlainLoginModule required
    username="user1"
    password="user1-secret" ;
};
```

Step 5) Kafka 서버 쉘 스크립트 설정 추가

■ kafka/bin/kafka-server-start.sh

```
export KAFKA_OPTS="-Djava.security.auth.login.config=/etc/kafka/kafka_server_jaas.conf"
```

Zookeeper 사용 시 추가 설정

만일 Kraft가 아닌 Zookeeper를 사용중이라면 다음과 같은 추가 설정이 필요하다

■ kafka/config/zookeeper.properties

```
zookeeper.sasl.client=true
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
requireClientAuthScheme=sasl
```

■ config/zookeeper_jaas.conf

```
Server {
    org.apache.kafka.common.security.plain.PlainLoginModule required
    username="admin"
```

```
    password="admin-secret"
    user_admin="admin-secret"
    user_user1="user1-secret" ;
};
```

7.3. SASL 진단 항목 List

분류	소분류	점검 항목
SASL	6.1	암호화 통신
	6.2	토큰 갱신
	6.3	SASL 인증 메시지 크기
	6.4	SASL 로그인 처리 시간 및 횟수

7.4. SASL 진단 항목

No 6.1	암호화 통신
개요	
점검 내용	■ SASL 통신 과정에서 암호화 통신 사용 여부 점검
보안위협	■ 암호화를 적용하지 않고 SASL/PLAIN 이 사용될 경우 민감 데이터 유출 가능성 증가
점검 대상 및 판단 기준	
점검 대상	■ Broker, kafka/config/kraft/server.properties ■ listener (default : null) : 리스너 설정 ■ advertised.listeners (default : null) : advertise 시 사용할 리스너 설정 ■ inter.broker.listener.name (default : null) : 내부 브로커 간 리스너 설정
판단 기준	양호 ■ SASL 이 PLAIN 일 경우 listener 와 inter.broker.listener.name 이 SASL_SSL 로 되어있거나 SASL_PLAINTEXT 를 사용하고 SASL/PLAIN 이외의 다른 매커니즘으로 되어있음
	취약 ■ SASL 이 PLAIN 임에도 listener 와 inter.broker.listener.name 이 SASL_PLAINTEXT 로 되어있거나 SASL_PLAINTEXT, SASL_SSL 이외에 다른 방식을 사용한다.
조치 방법	
kafka/config/kraft/server.properties 설정 확인	

- Listener, inter.broker.listener.name, advertised.listeners 설정이 SASL_PLAINTEXT 혹은 SASL/PLAIN 인 경우 SASL_SSL 로 설정되었는지 확인
- kafka/config/kraft/server.properties 에 아래 예시와 같이 작성

```
listeners=SASL_PLAINTEXT://:9092,CONTROLLER://:9093

inter.broker.listener.name=SASL_PLAINTEXT

advertised.listeners=SASL_PLAINTEXT://192.168.0.68:9092,CONTROLLER://192.168.0.68:9093
```

<table>
<tr><th>No 6.2</th><th>토큰 갱신</th></tr>
<tr><td colspan="2">개요</td></tr>
<tr><td>점검 내용</td><td>■ SASL 로그인 토큰이 적절한 시간 내에 적절한 횟수로 갱신되고 있는지 점검</td></tr>
<tr><td>보안위협</td><td>■ 불필요한 갱신 요청이 인증 서버를 과도하게 점유해 성능 저하 뿐 아니라 서비스 거부 공격으로 이어질 위험
■ 너무 긴 토큰 갱신으로 인해 잦은 서비스 중단 가능성</td></tr>
<tr><td colspan="2">점검 대상 및 판단 기준</td></tr>
<tr><td>점검 대상</td><td>■ Broker, kafka/config/kraft/server.properties
■ sasl.login.refresh.buffer.seconds (default : 300) : 로그인 토큰 갱신을 위해 남겨진 시간을 초 단위로 정의합니다. 이 시간 안에 토큰을 갱신합니다.
■ sasl.login.refresh.min.period.seconds (default : 60): 토큰 갱신 시 두 번의 연속 갱신 요청 간 최소 간격을 초 단위로 정의합니다.</td></tr>
<tr><td rowspan="2">판단 기준</td><td>양호
■ 갱신 버퍼 시간이 만료 전 충분한 여유를 두고 갱신 요청을 처리할 수 있는 상태
■ 연속 갱신 요청 간 최소 간격이 과도한 부하를 방지할 수 있는 값으로 설정되어 있는 상태</td></tr>
<tr><td>취약
■ 갱신 버퍼 시간이 짧아 인증 서버에 부하를 주거나 만료된 토큰으로 서비스 중단 가능성이 있는 상태
■ 간격이 이상으로 인증 서버에 과도한 요청을 보내거나 간격이 너무 길어 갱신 실패의 가능성이 높은 상태</td></tr>
<tr><td colspan="2">조치 방법</td></tr>
<tr><td colspan="2">적절한 설정값으로 설정
■ sasl.login.refresh.buffer.seconds 를 60 초 이상의 값으로 설정
■ sasl.login.refresh.min.period.seconds 를 30 초 이상, 300 초 미만의 값으로 설정
■ 그 외에 kafka 클러스터 환경 및 인증 서비스 응답 시간을 모니터링하여 적정 시간에 맞게 유동적으로 조정</td></tr>
</table>

No 6.3	SASL 인증 메시지 크기
개요	
점검 내용	■ SASL 메시지 사이즈의 적절성 검사
보안위협	■ 과도한 크기의 메시지는 공격자에 의해 서비스 거부 공격의 가능성 ■ 지나치게 작은 크기는 정상적인 클라이언트 요청도 거부하여 서비스 사용에 제약 가능성
점검 대상 및 판단 기준	
점검 대상	■ Broker, kafka/config/kraft/server.properties ■ sasl.server.max.receive.size (default : 1048576): 서버가 수신할 수 있는 최대 요청 크기를 정의
판단 기준	양호 ■ 크기 제한 설정값이 시스템 성능, 평균 메시지 크기에 적합하게 구성되어 과도한 메시지로 인한 자원 소진이나 정상 메시지 차단이 발생하지 않는 상태
	취약 ■ 값이 너무 클 경우 과도한 메시지로 인해 자원이 빠르게 소진되어 공격자가 서비스 거부 공격이 가능한 상태 ■ 값이 너무 작아 클라이언트의 정상 인증 메시지를 처리하지 못하는 상테
조치 방법	
고려 사항 ■ kafka/config/kraft/server.properties 에서 sasl.server.max.receive.size 의 값을 확인 ■ 클라이언트 인증 메시지는 일반적으로 작기 때문에 1~5MB 설정하는 것이 적절 ■ 실시간 모니터링을 통해 서비스에서 사용하는 인증 메시지의 크기를 분석, sasl.server.max.receive.size 의 값을 환경에 맞게 조정	

No 6.4	SASL 로그인 처리 시간 및 횟수
개요	
점검 내용	■ SASL 로그인 처리 시간 및 횟수가 적절한지 점검
보안위협	■ 클라이언트가 서버 연결을 설정하지 못해 인증 반복 실패 가능성 ■ 과도한 재시도 요청으로 공격자가 의도적을 서비스 거부 공격의 가능성
점검 대상 및 판단 기준	
점검 대상	■ Broker, kafka/config/kraft/server.properties ■ sasl.login.connect.timeout.ms (default : null) : 로그인 요청을 처리하기 위해 연결을 시도할 때의 최대 시간 정의 ■ sasl.login.retry.backoff.ms (default : 300): 로그인 요청이 실패한 후 재시도까지의 대기 시간 장정의 ■ sasl.login.retry.backoff.max.ms (default : 60000): 로그인 요청 실패 시 재시도 간 최대 대기 시간 정의
판단 기준	양호 ■ sasl.login.connect.timeout.ms 의 값이 클라이언트가 정상적으로 서버와 연결할 수 있는 상태 ■ 실패 시 재시도 간격이 적절히 설정되어 브로커의 인정 서비스에 과부하를 유발하지 않는 상태
	취약 ■ sasl.login.connect.timeout.ms 설정이 지나치게 낮아 연결 설정이 불가능하거나 서버가 과도한 요청을 처리하는 상태 ■ 과도한 재시도로 인해 브로커에 반복적으로 과부하게 발생하는 상태 ■ sasl.retry.backoff.max.ms 의 값이 너무 커 클라이언트의 인증 지연 및 서비스가 불가한 상태
조치 방법	
고려 사항 ■ sasl.login.connect.timeout.ms = 5000 ■ sasl.login.retry.backoff.ms = 500 ■ sasl.login.retry.backoff.max.ms = 10000 ■ 각 설정의 권장값은 다음과 같으며 모니터링을 통해 서비스 환경과 네트워크 상태를 조정할 것을 권장	

부록

<1. 요청 별 최소 필요 권한>

Protocol (API Key)	Operation	Resource	Note
PRODUCE (0)	Write	TransactionalId	Transaction ID가 설정된 Transaction Producer에 필요한 권한
PRODUCE (0)	IdempotentWrite	Cluster	Idempotent produce action 시 해당 권한 필요
PRODUCE (0)	Write	Topic	일반적인 Producer의 경우 해당 권한 필요
FETCH (1)	ClusterAction	Cluster	파티션 데이터를 패치하기 위해선 Cluster 리소스에 대한 ClusterAction 권한 필요
FETCH (1)	Read	Topic	일반적인 Consumer의 경우 각 파티션에 대한 READ 권한 필요
LIST_OFFSETS (2)	Describe	Topic	
METADATA (3)	Describe	Topic	
METADATA (3)	Create	Cluster	토픽 자동 생성 활성화 시, 브로커 측에서 Cluster 리소스에 대한 권한 확인. 없을 경우 하단의 Topic 리소스에 대한 권한 검사
METADATA (3)	Create	Topic	토픽 자동 생성 활성화 시 Cluster 리소스에 대한 권한이 존재하지 않을 시 Topic 리소스에 대한 권한 검사

LEADER_AND_ISR (4)	ClusterAction	Cluster	
STOP_REPLICA (5)	ClusterAction	Cluster	
UPDATE_METADATA (6)	ClusterAction	Cluster	
CONTROLLED_SHUTDOWN (7)	ClusterAction	Cluster	
OFFSET_COMMIT (8)	Read	Group	Group과 Topic에 대한 권한 필요
OFFSET_COMMIT (8)	Read	Topic	소비 프로세스의 일부이므로 읽기 작업에 대한 권한 필요
OFFSET_FETCH (9)	Describe	Group	Group 및 Topic에 대한 권한 필요
OFFSET_FETCH (9)	Describe	Topic	
FIND_COORDINATOR (10)	Describe	Group	해당 요청이 Group에서 온 경우
FIND_COORDINATOR (10)	Describe	TransactionalId	Transaction Producer가 Transaction Coordinator를 찾는 경우
JOIN_GROUP (11)	Read	Group	
HEARTBEAT (12)	Read	Group	
LEAVE_GROUP (13)	Read	Group	
SYNC_GROUP (14)	Read	Group	
DESCRIBE_GROUPS (15)	Describe	Group	
LIST_GROUPS (16)	Describe	Cluster	Cluster 리소스에 대한 권한 먼저 검사. 이후 하단과 같이 Group에 대한 Topic 권한 검사
LIST_GROUPS (16)	Describe	Group	Group 권한이 없는 경우 빈 응답 전송. 2.1부터 적용
SASL_HANDSHAKE (17)			

API_VERSIONS (18)			인증 이전에 발생하는 통신
CREATE_TOPICS (19)	Create	Cluster	Cluster 리소스에 대한 권한 우선 검사
CREATE_TOPICS (19)	Create	Topic	Topic에 경우 2.0부터 적용
DELETE_TOPICS (20)	Delete	Topic	
DELETE_RECORDS (21)	Delete	Topic	
INIT_PRODUCER_ID (22)	Write	TransactionalId	
INIT_PRODUCER_ID (22)	IdempotentWrite	Cluster	
OFFSET_FOR_LEADER_EPOCH (23)	ClusterAction	Cluster	Cluster 리소스 우선 검사
OFFSET_FOR_LEADER_EPOCH (23)	Describe	Topic	Topic에 경우 2.1부터 적용
ADD_PARTITIONS_TO_TXN (24)	Write	TransactionalId	Transaction 요청에만 적용. 해당 리소스 권한 검사 이후에 Topic에 대한 Write 권한 검사
ADD_PARTITIONS_TO_TXN (24)	Write	Topic	
ADD_OFFSETS_TO_TXN (25)	Write	TransactionalId	Transaction 요청에만 적용. 해당 리소스 권한 검사 이후 Group에 대한 Read 권한 검사
ADD_OFFSETS_TO_TXN (25)	Read	Group	
END_TXN (26)	Write	TransactionalId	
WRITE_TXN_MARKERS (27)	Alter	Cluster	
WRITE_TXN_MARKERS (27)	ClusterAction	Cluster	
TXN_OFFSET_COMMIT (28)	Write	TransactionalId	
TXN_OFFSET_COMMIT (28)	Read	Group	
TXN_OFFSET_COMMIT (28)	Read	Topic	

DESCRIBE_ACLS (29)	Describe	Cluster	
CREATE_ACLS (30)	Alter	Cluster	
DELETE_ACLS (31)	Alter	Cluster	
DESCRIBE_CONFIGS (32)	DescribeConfigs	Cluster	Broker Configs 요청 시 Cluster 수준 권한 확인
DESCRIBE_CONFIGS (32)	DescribeConfigs	Topic	Topic Configs 요청 시 Topic 수준 권한 확인
ALTER_CONFIGS (33)	AlterConfigs	Cluster	Broker Configs 변경 시 Cluster 수준 권한 확인
ALTER_CONFIGS (33)	AlterConfigs	Topic	Topic Configs 변경 시 Topic 수준 권한 확인
ALTER_REPLICA_LOG_DIRS (34)	Alter	Cluster	
DESCRIBE_LOG_DIRS (35)	Describe	Cluster	
SASL_AUTHENTICATE (36)			
CREATE_PARTITIONS (37)	Alter	Topic	
CREATE_DELEGATION_TOKEN (38)			위임 토큰 생성
CREATE_DELEGATION_TOKEN (38)	CreateTokens	User	사용자 리소스에 대한 위임 토큰 생성
RENEW_DELEGATION_TOKEN (39)			위임 토큰 갱신
EXPIRE_DELEGATION_TOKEN (40)			위임 토큰 만료
DESCRIBE_DELEGATION_TOKEN (41)	Describe	DelegationToken	위임 토큰 개요
DESCRIBE_DELEGATION_TOKEN (41)	DescribeTokens	User	사용자 리소스 위임 토큰 describe
DELETE_GROUPS (42)	Delete	Group	
ELECT_PREFERRED_LEADERS (43)	ClusterAction	Cluster	
INCREMENTAL_ALTER_CONFIGS (44)	AlterConfigs	Cluster	Broker Configs 변경 시

			Cluster 수준 권한 확인
INCREMENTAL_ALTER_CONFIGS (44)	AlterConfigs	Topic	Topic Configs 변경 시 Topic 수준 권한 확인
ALTER_PARTITION_REASSIGNMENTS (45)	Alter	Cluster	
LIST_PARTITION_REASSIGNMENTS (46)	Describe	Cluster	
OFFSET_DELETE (47)	Delete	Group	
OFFSET_DELETE (47)	Read	Topic	
DESCRIBE_CLIENT_QUOTAS (48)	DescribeConfigs	Cluster	
ALTER_CLIENT_QUOTAS (49)	AlterConfigs	Cluster	
DESCRIBE_USER_SCRAM_CREDENTIALS (50)	Describe	Cluster	
ALTER_USER_SCRAM_CREDENTIALS (51)	Alter	Cluster	
VOTE (52)	ClusterAction	Cluster	
BEGIN_QUORUM_EPOCH (53)	ClusterAction	Cluster	
END_QUORUM_EPOCH (54)	ClusterAction	Cluster	
DESCRIBE_QUORUM (55)	Describe	Cluster	
ALTER_PARTITION (56)	ClusterAction	Cluster	
UPDATE_FEATURES (57)	Alter	Cluster	
ENVELOPE (58)	ClusterAction	Cluster	
FETCH_SNAPSHOT (59)	ClusterAction	Cluster	
DESCRIBE_CLUSTER (60)	Describe	Cluster	
DESCRIBE_PRODUCERS (61)	Read	Topic	
BROKER_REGISTRATION (62)	ClusterAction	Cluster	
BROKER_HEARTBEAT (63)	ClusterAction	Cluster	
UNREGISTER_BROKER (64)	Alter	Cluster	

DESCRIBE_TRANSACTIONS (65)	Describe	TransactionalId	
LIST_TRANSACTIONS (66)	Describe	TransactionalId	
ALLOCATE_PRODUCER_IDS (67)	ClusterAction	Cluster	
CONSUMER_GROUP_HEARTBEAT (68)	Read	Group	
CONSUMER_GROUP_DESCRIBE (69)	Read	Group	
CONTROLLER_REGISTRATION (70)	ClusterAction	Cluster	
GET_TELEMETRY_SUBSCRIPTIONS (71)			
PUSH_TELEMETRY (72)			
ASSIGN_REPLICAS_TO_DIRS (73)	ClusterAction	Cluster	
LIST_CLIENT_METRICS_RESOURCES (74)	DescribeConfigs	Cluster	
DESCRIBE_TOPIC_PARTITIONS (75)	Describe	Topic	
SHARE_GROUP_HEARTBEAT (76)	Read	Group	
SHARE_GROUP_DESCRIBE (77)	Describe	Group	
SHARE_FETCH (78)	Read	Group	
SHARE_FETCH (78)	Read	Topic	
SHARE_ACKNOWLEDGE (79)	Read	Group	
SHARE_ACKNOWLEDGE (79)	Read	Topic	
INITIALIZE_SHARE_GROUP_STATE (83)	ClusterAction	Cluster	
READ_SHARE_GROUP_STATE (84)	ClusterAction	Cluster	
WRITE_SHARE_GROUP_STATE (85)	ClusterAction	Cluster	
DELETE_SHARE_GROUP_STATE (86)	ClusterAction	Cluster	
READ_SHARE_GROUP_STATE_SUMMARY (87)	ClusterAction	Cluster	