MQ Broker Security Guidelines

Kafka Security Settings Guide

Author	BoB 13th
Date of Preparation	2024.12.21

Table of Contents

1.Overview 4

1.1. Background 4

1.2. Diagnostic Checklist 5

2. Client 7

2.1. Overview 7

2.2. Client Diagnostic Checklists 7

2.3 Client Diagnostic Items 8

3. Broker 23

3.1. Overview 23

3.2. Broker Diagnostic Checklist 23

3.3. Broker Diagnostic Items 24

4. Topic 36

4.1. Overview 36

4.2. Topic Diagnostic Checklist 36

4.3. Topic Diagnostic Items 37

5. ACL (Access Control List) 43

5.1. Overview 43

5.2. Minimum required per request 43

5.3. ACL Diagnostic Checklist 43

5.4. ACL Diagnostic Items 44

6. SSL (Secure Sockets Layer) 54

6.1. Overview 54

6.2. SSL Application Guidelines 54

6.3. SSL Diagnostic Checklist 57

6.4. SSL Diagnostic Items 58

7. SASL (Simple Authentication and Security Layer) 62

7.1. Overview 62

7.2. SASL Configuration Guidelines 62

7.3. SASL Diagnostic Checklist 64

7.4. SASL Diagnostic Items 65

Supplementary Provisions 70

1.Overview

1.1. Background

Kafka is the first Message Queue Software developed by LinkedIn in 2011. Optimized for processing and transmitting large amounts of data in real time, it is a powerful platform that can handle numerous events and messages reliably and quickly. In addition, thanks to distributed architectures, the system can be used with high scalability and provides high availability and fault tolerance.

Currently, it has developed into an open-source project of the Apache Software Foundation and is used worldwide for various purposes such as message queuing, log management, data pipelines, and real-time analysis. Global foreign companies such as Netflix, Uber, Spotify, Airbnb, Goldman, and Slack are actively using the Kafka system, as well as famous domestic companies such as Naver, Woowa Brothers, Coupang, and Toss. As such, Kafka is establishing itself as an essential infrastructure for efficient data flow management and analysis in various companies and industries.

As many companies at home and abroad use Kafka, the impact of security risks is also very large. Kafka offers users hundreds of setup options, and these settings have a significant impact on the performance and security of the system. However, since Kafka comes with no security settings applied by default, any client within the same network band is free to access. As a result, there is a risk that malicious users can easily access or leak key information. In addition, incorrect manipulation of these settings in the process of maximizing system flexibility and optimizing performance can lead to various security incidents such as resource depletion, data leakage, permission bypass, denial of service attacks, etc. Therefore, it is essential for Kafka users to accurately understand and properly manage these Kafka's security settings.

This guide covers key security settings for operating Kafka safely and minimizing possible security threats. It focuses on the various security-related settings provided inside Kafka and provides appropriate setup methods and recommendations for reliably operating Kafka clusters.

1.2. Diagnostic Checklist

This guide is based on the diagnostic list for Kafka security checks. It presents diagnostic items for each resource such as Broker, Producer, and Consumer, and includes SASL and SSL setup methods and diagnostic items for them. Each item provides possible security risks, methods of setting up, and judgment criteria for the application of appropriate settings. Through this, the user reviews and applies each diagnostic item step by step to help the system operate safely. The table below summarizes the entire diagnostic list.

Classification	Sub-classification	Inspection Items
1. Client	1.1	Size of Produce message
	1.2	Produce message timeout
	1.3	Consumer message offset
	1.4	Consumer offset auto commit
	1.5	Request send/receive buffer
	1.6	Socket connection timeout
	1.7	Client DNS lookup
	1.8	Ensuring Message Delivery
2. Broker	2.1	Communication Protocol
	2.2	Socket communication
	2.3	Metadata Management
	2.4	Connection Recovery
	2.5	Change settings policy
	2.6	Maximum message size allowed for brokers
	2.7	Heartbeat between Brokers
3. Topic	3.1	Managing Access to Internal Topics
	3.2	Automatically generate topics
	3.3	Record batch size
	3.4	Topic Generation Policy
4. ACL	4.1	Principle of minimum authority
	4.2	Wildcard
	4.3	Super User
	4.4	Managing Cluster Resources
	4.5	Managing Describe privileges

	4.6	Unnecessary account management
	4.7	Allow access to Resource without ACL
5. SSL	5.1	Set client authentication request
	5.2	SSL Protocol
	5.3	Protocol mapping
	5.4	SSL Certificate Management
6. SASL	6.1	cryptographic communication
	6.2	Token Renewal
	6.3	SASL Authentication Message Size
	6.4	SASL login processing time and frequency

2. Client

2.1. Overview

Chapter 2 deals with clients who play a key role in sending and receiving messages in Kafka. Clients include Producer and Consumer, and they need to pay particular attention to security because they communicate directly with Kafka brokers in the process of exchanging data. A client may experience problems such as message loss, data leakage, and performance degradation if the design and settings are incorrect, which may impair stability for the entire Kafka system.

Therefore, this chapter introduces security and performance-related diagnostic items that must be considered when configuring the Kafka client.

2.2. Client Diagnostic Checklists

Classification	Sub-classification	Inspection Items
Client	1.1	Size of Produce message
	1.2	Produce message timeout
	1.3	Consumer message offset
	1.4	Consumer offset auto commit
	1.5	Request send/receive buffer
	1.6	Socket connection timeout
	1.7	Client DNS lookup
	1.8	Ensuring Message Delivery

2.3 Client Diagnostic Items

No 1.1	Size of Produce message
Overview	
Inspection Details	■ Check the appropriateness of message size-related settings when clients send messages to brokers
Security Threats	■ Excessive message size transfer can cause system resource depletion and service interruption risk and data processing failure ■ Too small batch size and message limit settings reduce transfer efficiency
Inspection Targets and Evaluation Criteria	
Inspection Targets	■ Producer client, broker, kafka/config/producer.properties ■ max.request.size (default: 1048576): the maximum size a producer can send to a broker once a request ■ buffer.memory (default: 33554432) : Total memory size that a producer can use to buffer records waiting to be sent to a server ■ batch.size (default: 16384): maximum size per batch when the producer handles messages as batches ■ message.max.bytes (default: 1048588) : Maximum size of a single batch
Evaluation Criteria	Good ■ Size limit settings are appropriately configured for system performance, average message size ■ batches.size and linger.ms set to efficiency for average message size and network bandwidth
	Vulnerable ■ Configured with excessive message sizes, leading to a high risk of resource exhaustion, or set too small, causing potential data processing failure ■ Mismatch between buffer.memory and message size settings, leading to the potential for resource overrun ■ Excessively small batch.size causing network bottlenecks
Recommended Actions	
Consideration ■ Set max.request.size and message.max.bytes consistently	

- Adjust buffer.memory to broker capacity to prevent excessive resource occupation
- Set batch.size appropriately to maintain efficient transfer
- Adjust linger.ms values to match message size

Configuration Examples

- Set kafka/config/producer.properties as in the following example

```
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 message timeout</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check the adequacy of producer timeout settings</td></tr>
<tr><td>Security Threats</td><td>■ If the timeout value is too short or too long, there is a possibility of delay.
■ Inadequate timeout settings can lead to increased service response times due to network bottlenecks or overloading of brokers
■ If encryption is not enabled, prolonged timeout increases the risk of data theft</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ 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 rowspan="2">Evaluation Criteria</td><td>Good
■ delivery.timeout.ms : Maximum time to report success or failure after the message request method send() call is returned. Limit the total time the record is delayed, the time it waits for the broker to verify, and the time allowed for retrievable transfer failures
■ linger.ms : Time to wait to bundle records into batches</td></tr>
<tr><td>Vulnerable
■ 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 colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Set delivery.timeout.ms values appropriately for network environment, broker processing time, message size
Maintaining consistency of settings</td></tr>
</table>

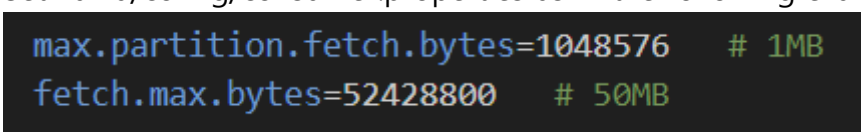
- Consistency by considering linger.ms and request.timeout.ms settings appropriately

Configuration Examples

- Set kafka/config/producer.properties as in the following example

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

<table>
<tr><th>No 1.3</th><th>Consumer message offset</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check key settings that control the size of the Consumer's data request</td></tr>
<tr><td>Security Threats</td><td>■ Too high setting values could lead to malicious consumers requesting excessive data, consuming the broker's resources.
■ If the setting value is inefficient, the processing speed may be lowered due to an increase in unnecessary network requests.</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Consumer client, kafka/config/consumer.properties
■ max.partition.fetch.bytes (default: 1048576) : Maximum data size that can be returned per partition
■ fetch.max.bytes (default: 52428800): Maximum data size to return for consumer data requests</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ It is tailored to the consumer's memory capacity and system needs.
■ Brokers and topics operating within set values</td></tr>
<tr><td>Vulnerable
■ Set too high or low to cause data processing failure or resource consumption
■ High settings can cause application failures due to memory overruns</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Set max.message.bytes and message.max.bytes consistently
■ Considering the memory capacity of consumer applications to prevent memory shortages

Configuration Examples
■ Set kafka/config/consumer.properties as in the following example
  </td></tr>
</table>

<table>
<tr><th>No 1.4</th><th>Consumer offset auto commit</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check key parameters for consumers to process messages and manage offsets</td></tr>
<tr><td>Security Threats</td><td>■ If the commit interval is too long, there is a possibility that messages will be consumed or lost in duplicate after uncommitted offsets in the event of an application failure.
■ Auto-commitment is less secure than manual-commitment, so malicious consumers are likely to distort the order of data processing.</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Consumer client, kafka/config/consumer.properties
■ enable.auto.commit(default: true): Set whether to commit offset automatically
■ auto.commit.interval.ms (default: 5000): Set the automatic offset commit interval
■ auto.offset.reset (default: late): Set the starting position to read the message if there is no offset</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ auto.commit.interval.ms is set at appropriate intervals
■ auto.offset.reset : Early or none applied to critical data processing
■ enable.auto.commit : state of fine control via manual commit with false</td></tr>
<tr><td>Vulnerable
■ auto.commit.interval.ms is too long or too short to cause performance degradation and data loss
■ If auto.offset.reset is set to late for critical data processing</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Set the appropriate value for consumer failures or network delays to avoid data loss or duplication because auto.commit.interval.ms is set too long apart
■ Prevent message loss when processing key data by using the Earlyest or none values
■ Recommended to use latest only for real-time data processing</td></tr>
</table>

Configuration Examples

- Create as below example in kafka/config/consumer.properties

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

<table>
<tr><th>No 1.5</th><th>Request send/receive buffer</th></tr>
<tr><th colspan="2">Overview</th></tr>
<tr><td>Inspection Details</td><td>■ TCP buffer management to use when transmitting/receiving clients</td></tr>
<tr><td>Security Threats</td><td>■ Inadequate buffer size settings may result in a vulnerability to Denial of Service (DoS) attack due to excessive system resource consumption
■ Too small a buffer can cause data loss when large amounts of messages arrive quickly</td></tr>
<tr><th colspan="2">Inspection Targets and Evaluation Criteria</th></tr>
<tr><td>Inspection Targets</td><td>■ Producer client, Consumer client, kafka/config/producer.properties, kafka/config/consumer.properties
■ send.buffer.bytes (default : 131072) : Size of TCP transmit buffer to use when transferring data
■ receive.buffer.bytes (default: 32768) : The size of the TCP receive buffer to use when reading data</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Properly sized transmit/receive buffers and running smoothly without overconsumption of resources or loss of performance or data</td></tr>
<tr><td>Vulnerable
■ Buffer size is too small or too large to cause problems such as data loss, performance degradation, and lack of resources</td></tr>
<tr><th colspan="2">Recommended Actions</th></tr>
<tr><td colspan="2">Consideration
■ receive.buffer.bytes and send to match the network environment and traffic patterns of the system.buffer.bytes settings
■ OS defaults apply when setting this setting to -1
■ If OS defaults are optimized and the system network environment is taken into account, reflect that setting

Configuration Examples
■ Write in kafka/config/consumer.properties and kafka/config/producer.properties as below example</td></tr>
</table>

```
send.buffer.bytes=-1  # OS 기본값 사용
receive.buffer.bytes=262144  # 256KB
```

No 1.6	Socket connection timeout
Overview	
Inspection Details	■ Set the timeout time required when the client establishes a TCP connection with the broker
Security Threats	■ If the timeout setting is inefficient, there is a possibility that abnormal access blocking will fail. ■ Excessive waiting for unusual connection requests may cause the service to consume resources and cause a denial attack
Inspection Targets and Evaluation Criteria	
Inspection Targets	■ Producer client, Consumer client, kafka/config/producer.properties, kafka/config/consumer.properties ■ socket.connection.setup.timeout.max.ms (default: 30000): the amount of time a producer waits when attempting to connect to a broker ■ socket.connection.setup.timeout.ms (default: 10000): Maximum amount of time a client waits to try to connect
Evaluation Criteria	Good ■ The appropriate timeout time is set for your system environment.
	Vulnerable ■ Conditions that can be wasted or delayed due to too short or long timeout settings
Recommended Actions	

Consideration

- Set up for network delay and reliability
- Increased timeout in high network latency environments and reduced speed when quick response is required
- Prove connectivity performance and reliability with sufficient testing

Configuration Examples

- Write in kafka/config/consumer.properties and kafka/config/consumer.properties as below example

```
socket.connection.setup.timeout.max.ms=40000  # 40초
socket.connection.setup.timeout.ms=15000  # 15초
```

No 1.7	Client DNS lookup
Overview	
Inspection Details	■ Determines the DNS lookup mode that the client uses to resolve the broker's hostname.
Security Threats	■ A malicious DNS server modulates the DNS cache, leading to incorrect IP and possibly DNS spoofing. ■ Incorrect DNS settings may lead to connection failure, destabilizing the connection with the broker
Inspection Targets and Evaluation Criteria	
Inspection Targets	■ Producer client, Consumer client, kafka/config/producer.properties, kafka/config/consumer.properties ■ client.dns.lookup (default: use_all_dns_ips): Set DNS lookup mode when checking broker hostname - use_all_dns_ips : Attempt to connect to all returned IP addresses sequentially during DNS lookup - reserve_canonical_bootstrap_servers_only: interpret the address of the bootstrap server as one full name
Evaluation Criteria	Good ■ use_all_dns_ips or reserve_canonical_bootstrap_servers_only is set up for your network environment and is set securely ■ Status that a trusted DNS server is set up and DNS security audits are performed periodically
	Vulnerable ■ Using DNS server with improper DNS settings or untrusted
Recommended Actions	
Consideration ■ use_all_dns_ips: Useful when flexibility is required in a network environment, but if a DNS server is tampered with, it can attempt to connect to a malicious IP, so use a trusted DNS server and enhance DNS security ■ reserve_canonical_bootstrap_servers_only: used only in the bootstrap phase, interpreting the server name as a single formal name. It can increase network stability, but it can limit the	

flexibility of multipath connections. It is important to verify the reliability of the DNS server when setting this value.

Configuration Examples

- Write in kafka/config/consumer.properties and kafka/config/producer.properties as below example

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

<table>
<tr><th>No 1.8</th><th>Ensuring Message Delivery</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check settings to ensure that messages are sent exactly once, or processed in order without duplication</td></tr>
<tr><td>Security Threats</td><td>■ Critical data processing can cause message order mismatch issues for systems that require order guarantees
■ Message duplication can cause data integrity issues</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Producer client, kafka/config/producer.properties
■ enable.idempotence (default: true): Ensure that the same message is not duplicated when the producer sends the message
■ max.in.flight.request.per .connection (default: 5) : Maximum number of requests a client can simultaneously send unacknowledged requests on a connection.</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ A state in which enable.idempotence is set to true, preventing duplicate messages and ensuring the accuracy of the data.Status that a trusted DNS server is set up and DNS security audits are performed periodically
■ max.in.flight.request.per.connection is set to 5 or less, retries is set to 0 or greater, and aks is set to all, so the feature is enabled without conflict.</td></tr>
<tr><td>Vulnerable
■ A condition where enable.idempotence is set to false so that messages can be logged in duplicate, which poses a risk of compromising data consistency and accuracy
■ max.in.The enable.idempotence feature is disabled due to conflicting settings, such as when flight.request.per .connection exceeds 5 and aks are not all.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ enable.idempotence enable
■ max.in.Set the value of flight.request.per .connection to 5 or less
■ Set retries value to 0 or greater</td></tr>
</table>

- Set aks value to all
- Check enable.idempotence to be enabled safely by applying the above settings
- After setup, verify connectivity performance and reliability with sufficient testing

Configuration Examples

- Write in kafka/config/consumer.properties and kafka/config/producer.properties as below example

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

3. Broker

3.1. Overview

Chapter 3 deals with brokers, the most important component of Kafka. Brokers mediate data transmission and reception between producers and consumers and store and manage data within the cluster. Brokers are important factors in determining the stability and performance of Kafka clusters, so management in various aspects, including communication protocol management and metadata, is necessary.

If the broker is not working properly, problems such as data processing delay, message loss, and system failure can occur, which greatly affects the client and the entire Kafka system. Therefore, this chapter introduces key diagnostic items to maintain the security and stability of Kafka brokers.

3.2. Broker Diagnostic Checklist

Classification	Sub-classification	Inspection Items
Broker	2.1	communication protocol
	2.2	Socket communication
	2.3	Metadata Management
	2.4	Connection Recovery
	2.5	Change settings policy
	2.6	Maximum message size allowed for brokers
	2.7	Heartbeat between Brokers

3.3. Broker Diagnostic Items

<table>
<tr><th>No 2.1</th><th>communication protocol</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check the reliability of the protocol and the use of the latest version for communication between brokers</td></tr>
<tr><td>Security Threats</td><td>■ Possible exposure of data to plaintext on the network when using PLAINTEXT</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>Consumer client, Broker, kafka/config/consumer.properties, kafka/config/kraft/server.properties
■ security.inter.broker.protocol (default: null) : the security protocol used in broker-to-broker communication
■ inter.broker.protocol.version (default: 3.8-IV0A): The version of the security protocol used in the broker-to-broker communication.</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ The security.inter.broker.protocol value is set to a value other than PLAINTEXT.
■ The inter.broker.protocol.version value is set to the most recent or undetected version.</td></tr>
<tr><td>Vulnerable
■ The security.inter.broker.protocol value is set to PLAINTEXT and no encryption is taken during the communication process.
■ The value of inter.broker.protocol.version is set to the version where the vulnerability was detected.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">security.inter.broker.protocol 변경
■ security.inter.broker.protocol is set to PLAINTEXT by default
■ Set the corresponding value by selecting the appropriate secure communication according to the data
inter.broker.protocol.version 설정
■ All brokers in the cluster must have the same value</td></tr>
</table>

- Be sure to update your current protocol version carefully to avoid compatibility issues in communications between brokers

<table>
<tr><th>No 2.2</th><th>Socket communication</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check socket communication-related settings to ensure that the transmit/receive buffer size and client request size are properly set</td></tr>
<tr><td>Security Threats</td><td>■ Inadequate transmit/receive buffer size setting, network bottlenecks or data loss are likely
■ Possibly wasting memory resources if buffer size or request size is set excessively relative to actual traffic</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ socket.request.max.bytes (default: 104857600): Maximum size of requests a client can send to a broker
■ socket.send.buffer.bytes (default: 102400): Size of the transmit buffer used by the socket server
■ socket.receive.buffer.bytes (default: 102400): the size of the receiving buffer used by the socket server</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Socket send and receive buffer values are set to the appropriate size for network traffic and bandwidth
■ The maximum request size is less than javaheap and greater than or equal to the message.max.bytes value.</td></tr>
<tr><td>Vulnerable
■ Transmit/receive buffer size is too small to cause network bottlenecks or set larger than necessary to waste memory
■ The maximum request size is greater than javaheap or less than message.max.bytes.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Test to match the network environment and traffic patterns of the system to derive and set the appropriate size of the transmitting and receiving buffers
■ Set the maximum request size to a value greater than or equal to the message size in use by the cluster</td></tr>
</table>

Configuration Examples

- Write in kafka/config/kraft/server.properties as below example

```
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)
```

<table>
<tr><th>No 2.3</th><th>Metadata Management</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check the proper settings for recovery of critical information, survival, and idle time for metadata</td></tr>
<tr><td>Security Threats</td><td>■ Old metadata can cause network delays as well as concurrency issues
■ Too often updating metadata requires a lot of resources due to the strain on the network</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ metadata.max.age.ms (default: 300000): Maximum survival time for metadata
■ metadata.max.idle.ms (default: 300000): Maximum idle time since metadata was most recently updated
■ metadata.recovery.strategy (default: none): Set up control of metadata recovery strategies when all brokers are unavailable</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Metadata survival time and idle time are properly set
■ Status set to the right metadata recovery strategy for your environment</td></tr>
<tr><td>Vulnerable
■ Metadata survival and idle times are too short or long
■ Metadata recovery strategies for your environment are not properly established</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">metadata.max.age.ms 설정
■ Keep up-to-date metadata and comply with security policies
metadata.max.idle.ms
■ Set the metadata update to the appropriate cycle in a cluster environment so that it does not age

appropriate metadata.recovery.strategy settings
■ NONE: Used in environments where brokers are less likely to change and need to quickly detect problems in the event of a failure because they do not attempt client recovery.</td></tr>
</table>

- REBOOTSTRAP: Attempt to connect to a new broker with the bootstrap process. Use for connections that are frequently changed by brokers or are likely to reactivate clients after being inactive for a long time

<table>
<tr><th>No 2.4</th><th>Connection Recovery</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check if the network connection is lost and the communication recovery settings between the broker and the client are properly established.</td></tr>
<tr><td>Security Threats</td><td>■ In case of continuous connection failure, network logs and broker IP ports are revealed, possibly exploiting attackers</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ reconnect.backoff.max.ms (default: 1000): Maximum time to wait when reconnecting to the host
■ reconnect.backoff.ms (default:50) : Default time to wait when reconnecting to host
■ retry.backoff.max.ms (default: 1000): Maximum time to wait upon re-request from host
■ retry.backoff.ms (default: 100): Default time to wait upon re-request from host</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Maximum value restricted to prevent repeated connection/request attempts from overloading brokers
■ Proper use of resources by setting appropriate initial values</td></tr>
<tr><td>Vulnerable
■ Conditions that are not moderately high or too high inappropriate maximums can lead to overload of brokers
■ Connection/request attempts made too short an interval, leading to resource abuse</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ If the reconnection waiting time is short, there is a possibility of waste of broker resources and DoS.
■ Longer client reconnection delays may result in lower service availability</td></tr>
</table>

- Short wait times for re-request can degrade broker processing performance due to request congestion
- Longer re-request wait times may result in lower service availability due to client request delays

appropriate settings

- It is recommended to fine-tune based on cluster traffic, authentication methods, and network reliability, considering security/stability balance in a test environment.
- Recommend optimization progress by analyzing connection failure frequency and re-request traffic patterns

Configuration Examples

- Write in kafka/config/kraft/server.properties as below example

```
# 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	Change settings policy
Overview	
Inspection Details	■ When you change the settings of a resource, check that the policies defined by the alter.config.policy.class.name settings are being applied.
Security Threats	■ Inadequate configuration changes may waste resources or degrade system performance ■ Undiscriminatory changes in critical security settings could potentially worsen the security of the cluster
Inspection Targets and Evaluation Criteria	
Inspection Targets	■ Broker, kafka/config/kraft/server.properties ■ alter.config.policy.class.name (default: null): Specify a policy class to control changes in settings for resources
Evaluation Criteria	Good ■ alter.config.policy.class.name settings are enabled and policy classes are functioning properly
	Vulnerable ■ Setting is disabled, policy class is not defined, or implemented incorrectly
Recommended Actions	

Consideration

■ Prepare appropriate policies for setting change control

Configuration Examples

Step 1) Implement the AlternateConfigPolicy interface as shown in the example below

■ The following example implements a policy class that requires data retention time (retention.ms) to be at least 1 hour.

```
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
{
```

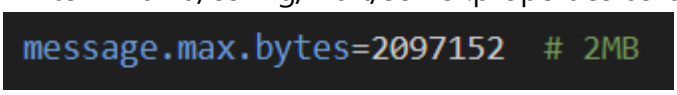
```
        String retentionMs = requestMetadata.configs().get("retention.ms");
        if (retentionMs != null && Long.parseLong(retentionMs) < 3600000) {   // 1 hour =
3600000ms
            throw new PolicyViolationException("Retention time must be at least 1 hour.");
        }
    }
    @Override
    public void configure(Map<String, ?> configs) {
//        Implement if additional settings are required
    }
    @Override
    public void close() {
//        Implement when resources need to be cleaned up
    }
}
```

Step 2) Set policy classes in kafka/config/kraft/server.properties

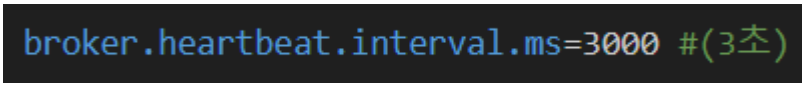
- Set as shown in the example below

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

<table>
<tr><th>No 2.6</th><th>Maximum message size allowed for brokers</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Set the maximum size of messages that the Kafka broker can handle</td></tr>
<tr><td>Security Threats</td><td>■ Too loud messages can cause overload and denial of service between brokers and clients
■ Too small messages have the potential to use network and system resources inefficiently</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ message.max.bytes (default: 104858) : Maximum message placement size allowed by broker
■ This size is based on the size after compression when compression is enabled.</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ message.max.bytes values are set for your system environment, and size and batch size are being managed efficiently</td></tr>
<tr><td>Vulnerable
■ The setting is set too large or too small to degrade or consume excessive resources.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Properly limit message size to suit your system environment
■ Verify that the message size that can be processed by the cluster meets application requirements

Configuration Examples
■ Write in kafka/config/kraft/server.properties as below example
  </td></tr>
</table>

<table>
<tr><th>No 2.7</th><th>Heartbeat between Brokers</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check items that set the heartbeat cycle between brokers</td></tr>
<tr><td>Security Threats</td><td>■ Too short heartbeat intervals increase unnecessary network traffic between brokers and increase the likelihood of system load failures
■ An excessively long heartbeat interval is a risk of service failure due to late detection of connectivity between brokers.</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ broker.heartbeat.interval.ms (default: 2000): Set the heartbeat interval between brokers</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Set values are set appropriately in the network environment and heartbeat intervals are not too short or long to ensure broker stability.</td></tr>
<tr><td>Vulnerable
■ Network load or service failure detection is late because the setting value is set too short or too long.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Set up appropriately considering network environment
■ High latency environments require increasing values to maintain reliable connectivity
■ Consider reducing network load by increasing the value appropriately if there are many brokers in the cluster

Configuration Examples
■ Write in kafka/config/kraft/server.properties as below example
  </td></tr>
</table>

4. Topic

4.1. Overview

Chapter 4 deals with Topic, the basic unit that defines Kafka's data flow. A topic is a logical unit in which data is stored and exchanged in Kafka, where the producer sends the message and the consumer consumes the message. Without proper topic management, risks such as message loss, data consistency problems, and performance degradation may occur.

Kafka topics play an important role in efficient distribution and management of message streams, especially partitioning, replication, and storage settings, which greatly affect the performance and security of the topic. This chapter introduces security management measures and diagnostic items to safely and effectively operate Kafka topics.

4.2. Topic Diagnostic Checklist

Classification	Sub-classification	Inspection Items
Topic	3.1	Managing Access to Internal Topics
	3.2	Automatically generate topics
	3.3	Record Placement Size (Topic)
	3.4	Topic Generation Policy

4.3. Topic Diagnostic Items

<table>
<tr><th>No 3.1</th><th>Managing Access to Internal Topics</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Controls whether consumers can read internal topics (for example, __consumer_offsets)</td></tr>
<tr><td>Security Threats</td><td>■ Internal topics contain sensitive metadata, such as cluster status or offset information.
■ Manage sensitive information from being leaked to the outside world</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Consumer client, kafka/config/consumer.properties
■ exclude.internal.topics (default: true): Set whether the Consumer can read internal topics</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Exclude.internal.topics = true, consumer is unable to access internal topics</td></tr>
<tr><td>Vulnerable
■ Exclude.internal.topics = false, or allowed access to internal topics</td></tr>
<tr><td colspan="2">Recommended Actions</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 Configuration
■ Set up ACLs to clearly restrict client access to internal topics</td></tr>
</table>

No 3.2	Automatically generate topics
Overview	
Inspection Details	■ When a client subscribes or allocates a topic that does not exist in the broker, the broker determines whether to automatically generate the topic.
Security Threats	■ If allow.auto.create.topics & auto.create.topics.enable is set to true, malicious users can create unnecessary topics ■ Automatically generated unnecessary topics may waste cluster resources and affect performance
Inspection Targets and Evaluation Criteria	
Inspection Targets	■ Consumer client, Broker, kafka/config/consumer.properties, kafka/config/kraft/server.properties ■ allow.auto.create.topic (default: true): Set whether a consumer automatically generates a topic when subscribing to a topic that does not exist in the broker ■ auto.create.topics.enable (default: true): Enables a client to automatically generate a topic when it tries to subscribe to or assign a topic that does not exist in the broker.
Evaluation Criteria	Good ■ allow.auto.create.topics and auto.create.topics.enable are set to false and the client does not waste resources by creating unnecessary topics
	Vulnerable ■ a condition where allow.auto.create.topics and auto.create.topics.enable are set to true and the client is likely to waste resources or cause difficulties in security management by creating unnecessary topics
Recommended Actions	
allow.auto.create.topics, auto.create.topics.enable 설정 ■ These options are set to true by default ■ By default, it is safe to disable it, but consider the development and testing environment or the environment in which the topic needs to be quickly processed to determine whether to activate the option ■ kafka/config/consumer.properties 에서 allow.auto.create.topics 설정 ■ kafka/config/kraft/server.properties 에서 auto.create.topics.enable 설정	

Configuration Examples

- Write in kafka/config/consumer.properties as below example

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

- Write in kafka/config/kraft/server.properties as below example

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

<table>
<tr><th>No 3.3</th><th>Record Placement Size (Topic)</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check maximum record batch size settings allowed by Kafka</td></tr>
<tr><td>Security Threats</td><td>■ Too loud messages can cause overload and denial of service between brokers and clients
■ Too small messages have the potential to use network and system resources inefficiently</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Topic
■ max.message.bytes (default: 1048588) : Maximum record batch size allowed by Kafka
■ This size is based on the size after compression when compression is enabled.</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ The max.message.bytes value is set for the system environment, and message size and batch size are being managed efficiently.</td></tr>
<tr><td>Vulnerable
■ The setting is too large or too small to consume performance or excessive resources.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Properly limit message size to suit your system environment
■ Set the max.message.bytes (topic) and message.max.bytes (Broker) values the same to prevent errors during data transfer

Configuration Examples
■ Can be set, as in the example of the command below
<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	Topic Generation Policy
Overview	
Inspection Details	■ When creating a topic, check that the policies defined through the create.topic.policy.class.name settings are being applied
Security Threats	■ Without proper topic generation policies, malicious users can generate inappropriate topics, which can lead to incidents such as waste of leases, performance degradation, etc. ■ Risk of data loss if a topic with too few replicas is created because there is no topic creation policy
Inspection Targets and Evaluation Criteria	
Inspection Targets	■ Broker, kafka/config/kraft/server.properties ■ create.topic.policy.name (default: null): The class name that defines the policy for the topic generation request.
Evaluation Criteria	Good ■ create.topic.policy.name settings are enabled and policy classes are functioning properly
	Vulnerable ■ Settings are disabled, policy classes are not defined, or incorrectly implemented
Recommended Actions	

Consideration

■ Set for topic name, number of replications, and number of partitions

Configuration Examples

Step 1) Implement the Create TopicPolicy interface as shown in the example below

■ The following example shows an interface that verifies the minimum number of partitions:

```
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 {
```

```
//          Example: Policy that the number of partitions in a topic must be at least 3
        if (requestMetadata.numPartitions() < 3) {
            throw new PolicyViolationException("Topic must have at least 3 partitions.");
        }
//          Additional verification logic can be implemented.
    }
    @Override
    public void configure(Map<String, ?> configs) {
//          Implement logic to handle settings for use in policy classes, if necessary
    }
    @Override
    public void close() {
//          Implementing Resource Release Logic at the end of a Policy
    }
}
```

Step 2) Set policy classes in kafka/config/kraft/server.properties

- Set as shown in the example below

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

5. ACL (Access Control List)

5.1. Overview

Chapter 5 deals with Access List Control (ACL), which is responsible for "authorization" in kafka's security model. ACLs perform data access control by defining user or application rights for Kafka resources (e.g., topics, consumer groups, etc.). This ensures the data confidentiality and integrity of the Kafka cluster.

Proper ACL setting is critical to prevent access by unauthorized users and to prevent unintended data changes or deletion. In particular, incorrect ACL configuration can lead to serious security issues such as data exposure and permission bypass. This chapter provides diagnostic items to safely apply and manage Kafka ACLs.

5.2. Minimum required per request

When applying ACLs in Kafka, following the principle of minimum authority is essential to enhance security and prevent unintended data access and changes. By granting only the minimum necessary authority for each request or action, the security risk caused by overauthorization may be reduced and the stability of the Kafka cluster may be maintained.

Appendix 1 is a list of the minimum rights required for each request of kafka. When designing or managing ACLs, it is recommended to refer to the list and set it up safely.

5.3. ACL Diagnostic Checklist

Classification	Sub-classification	Inspection Items
ACL	4.1	Principle of minimum authority
	4.2	Wildcard
	4.3	Super User
	4.4	Managing Cluster Resources
	4.5	Managing Describe privileges
	4.6	Unnecessary account management
	4.7	Allow access to Resource without ACL

5.4. ACL Diagnostic Items

<table>
<tr><th>No 4.1</th><th>principle of minimum authority</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Review ACL settings to ensure that the permissions granted to users and applications conform to the minimum privilege principle.</td></tr>
<tr><td>Security Threats</td><td>■ Unnecessary authorization can cause damage such as data leakage, deletion, and tampering due to malicious users or internal errors.</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ User and application permissions managed through ACLs in Kafka</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Only the minimum permissions required for users and applications are granted
■ Set to meet the above <Table 1. Minimum Rights Management Criteria></td></tr>
<tr><td>Vulnerable
■ The user or application is granted unnecessary privileges.
■ If the above <Table 1. Minimum Rights Management Criteria> are set to not meet the above <Table 1.></td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Use the kafka/bin/kafka-acl.sh script to check, grant, delete, and modify ACLs.

step 1) Check ACLs
■ All ACLs registered in the system can be checked by the command below.
<code>kafka-acls.sh --bootstrap-server <broker-url> --list</code>

step 2) Modifying ACLs
■ <Appendix 1. Confirmation of users who are set up not to meet the minimum authority required for each request.
■ If not, remove or modify the ACL with the command below.</td></tr>
</table>

Delete ACLs

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

<table>
<tr><th>No 4.2</th><th>Wildcard</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Verify that permissions with wildcard or pattern matching are set appropriately in ACL settings</td></tr>
<tr><td>Security Threats</td><td>■ Excessive use of wildcards may allow access to unintended resources and risk data leakage, deletion, or tampering.
■ Resource access can be overstretched with unnecessary privileges</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Items with resource patterns set to prefixed or wildcard in ACLs</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Used only for resources requiring wildcard and pattern matching, restricted to prevent unintended resource access
■ If the resource pattern is set to prefixed, the prefix or suffix is clearly and restrictively set.</td></tr>
<tr><td>Vulnerable
■ Wildcard and pattern matching is used for more than necessary privileges to allow unnecessary resource access.
■ Other resources are unintentionally authorized because there are no clear restrictions on ACL settings.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Minimize permissions
■ Minimize the use of wildcard and pattern matching and explicitly authorize certain resources
■ Example: Set permissions by specifying specific topic resources instead of wildcards
<code>kafka-acls.sh --add --allow-principal User:producer1 --operation WRITE --topic my-topic</code>

Restrictive pattern matching
■ When setting resource patterns to prefixed, use the prefixes carefully designed and restricted
<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">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Verify that there are too many users defined in the super.user settings, and verify that only the minimum required users are set to have super user privileges.</td></tr>
<tr><td>Security Threats</td><td>■ Unnecessarily granted superuser access to malicious attacks or all resources in the cluster, resulting in serious security threats such as data leakage, tampering, and deletion.</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ List of users defined in super.user settings</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ The user defined in super.users is minimized, and only those essential for cluster management have super user privileges.
■ Users with super user privileges check regularly and modify privileges if necessary</td></tr>
<tr><td>Vulnerable
■ The super.user has an excessive number of users and is granted unnecessary privileges.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">super.user review
■ Check super.user settings in the kafka/config/kraft/server.properties file and remove unnecessary users or privileges
■ Check super.user entry in kafka/config/kraft/server.properties as below example
<pre>super.users=User:admin
super.users=User:alice
super.users=User:bob
super.users=User:carol
super.users=User:dave</pre>
ACL Segmentation of Privileges</td></tr>
</table>

- Minimize the list of super users and break down permissions through ACLs as much as possible to grant only the required permissions

<table>
<tr><th>No 4.4</th><th>Managing Cluster Resources</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check cluster resource permissions granted to users</td></tr>
<tr><td>Security Threats</td><td>■ Granting unnecessary users permission to the cluster can lead to malicious behavior, such as changing system settings or over-creating resources.</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Users and Applications with permissions to cluster resources</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Granted to the least number of users who need permission to cluster resources, no unnecessary permissions granted</td></tr>
<tr><td>Vulnerable
■ Excessive number of users are granted cluster resource permission, which risks system configuration changes or resource creation abuse</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Minimize permissions
■ Minimize permission to cluster resources, empowering only required users

Review and periodic inspection of permissions
■ Review cluster resource permissions regularly to ensure unnecessary permissions are not granted
kafka-acl.sh --list --resource-cluster
■</td></tr>
</table>

<table>
<tr><th>No 4.5</th><th>Managing Describe privileges</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Describe permissions provide permission to read metadata for resources, primarily to query the status or property information of resources
■ This privilege is automatically granted if READ, WRITE, DELETE, ALTER privileges are granted (DESCRIBE_CONFIG is granted if ALTER_CONFIG is granted)</td></tr>
<tr><td>Security Threats</td><td>■ This allows authorized users to view metadata for resources, and when that information is exposed, sensitive data or information about the structure of the system can be leaked.
■ If this privilege is abused, details about the state of the resource can be found to identify vulnerabilities in the service or gather information for attacks</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ READ, WRITE, DELETE, ALTER, ALTER_CONFIG authorized users</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Describe privileges are not unnecessarily granted for sensitive resources, and privileges are carefully managed</td></tr>
<tr><td>Vulnerable
■ When READ, WRITE is granted, DESCRIBE is automatically granted, providing excessive information access to sensitive resources</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Minimize Describe permissions so that unnecessary users are not granted them
■ Recognize that when granting READ, WRITE, DELETE, ALTER, ALTER_CONFIG, DESCRIBE privileges are granted together and set permissions</td></tr>
</table>

<table>
<tr><th>No 4.6</th><th>Unnecessary account management</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check to identify and delete long-disused or unnecessary accounts</td></tr>
<tr><td>Security Threats</td><td>■ Possibility of an attacker exploiting an unnecessary account if it remains granted dangerous privileges
■ If an unused account has access, there is a possibility that an external attacker or malicious insider could exploit it to invade the system.</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Kafka user account, broker log</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Unnecessary accounts are disabled or deleted
■ Status with minimum user required privileges
■ When account reviews are done periodically</td></tr>
<tr><td>Vulnerable
■ An account that has been inactive for a long time or an account that is not needed is still active.
■ No periodic account review and cleanup</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Log Analysis
■ Track your activity in kafkalog and identify old or inactive accounts
<code>grep "User:" /path/to/kafka/logs/server.log</code>
■ Additionally, ACLs are clearly established to enhance security.

Analyzing the Offset of the Consumer Group
■ kafka-consumer-groups.Use the sh command to view the last offset information for a particular Consumer Group
<code>bin/kafka-consumer-groups.sh --bootstrap-server <broker_address> --describe --group <consumer_group_name></code></td></tr>
</table>

Periodic Account Checks

■ In addition to the above methods, periodic account checks are required

<table>
<tr><th>No 4.7</th><th>Allow access to Resource without ACL</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Determines the default access control method for resources with no ACL defined</td></tr>
<tr><td>Security Threats</td><td>■ Access to ACL-free resources unprotectedly exposes sensitive data, Topic, and other resources
■ Increased unintended data leakage and unauthorized access to systems</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ allow.everyone.if.no .acl.found=false with appropriate ACLs defined for all resources
■ ACL is not properly defined even if the setting is false</td></tr>
<tr><td>Vulnerable
■ Resource is set to allow.everyone.if.no .acl.found=true with no ACL set
■ ACL is not properly defined even if the setting is false</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">allow.everyone.if.no.acl.found 비활성화
■ Add code below to kafka/config/kraft/server.properties
kafka/config/kraft/broker.properties = false
■ Set ACLs appropriately to ensure that no ACLs are configured and clear access control settings</td></tr>
</table>

6. SSL (Secure Sockets Layer)

6.1. Overview

Chapter 6 deals with SSL, an encryption system applicable to Kafka. SSL is a protocol to encrypt data transmission on the network to prevent security threats such as man-in-the-middle attacks, data eavesdropping, and data modulation. Kafka can apply SSL to ensure the confidentiality and integrity of data transmission, enabling secure communication between clients and brokers. It also supports mutual authentication between clients and brokers, preventing untrusted users or systems from accessing the network. This can further enhance the security of the Kafka cluster.

This chapter describes the SSL application method and setup procedures step by step and provides a checklist to check the main diagnostic items and correct settings. This helps Kafka users effectively configure SSL and build a secure data transmission environment.

6.2. SSL Application Guidelines

Below is an example of applying SSL to Kafka. These guidelines are merely examples, and we recommend that they be applied well to the system environment or considerations to build a secure data transfer environment.

Step 1) Generating SSL certificates

Using Open SSL to Create SSL Certificates

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

- ca-key : the file to store the private key
- ca-cert : the file to store the generated certificate
- 365 : Validity of certificate

Step 2) Create a trust store

- The Trust Store is a trusted certificate store role when making SSL communications and adds CA certificates to it.

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

- truststore.jks : truststore file
- ca-cert : CA certificate generated in previous step

Step 3) Create a keystore

Create a keystore for Kafka brokers.Create a keystore that contains private keys and server certificates during this process

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

- keystore.jks : keystore file
- RSA:key algorithms to use
- SAN=dns:localhost : Additional name for SSL certificates

Step 4) Generate and Sign CSR(Certificate Signing Request)

Create a CSR on the keystore and sign it as a CA certificate to generate the broker's final certificate

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

Sign CSR with CA certificate

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

Step 5) Add to Certificate Keystore

Add signed certificates to the keystore. Kafka brokers can use the certificate to establish SSL/TLS connections

```
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) Set up the broker

■ 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 Diagnostic Checklist

Classification	Sub-classification	Inspection Items
SSL	5.1	Set client authentication request
	5.2	SSL Protocol
	5.3	Protocol mapping
	5.4	SSL Certificate Management

6.4. SSL Diagnostic Items

<table>
<tr><th>No 5.1</th><th>Set client authentication request</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check settings for authentication requests from clients</td></tr>
<tr><td>Security Threats</td><td>■ If authentication is not required, untrusted clients can access the kafka cluster</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ ssl.client.auth (default: none): Set whether client authentication is requested or not
■ required: client authentication required
■ requested: client authentication request
■ none : No client authentication</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Client authentication is enabled and a certificate must be provided to access the Kafka broker.</td></tr>
<tr><td>Vulnerable
■ Client authentication is optional or the authentication request is disabled.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">ssl.client.auth 설정
■ Set ssl.client.auth to required so that all clients must provide a certificate to access the broker
■ kafka/config/kraft/server.properties
<code>ssl.client.auth=required</code></td></tr>
</table>

<table>
<tr><th>No 5.2</th><th>SSL Protocol</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check that SSL protocol settings are configured with the latest and recommended protocols
■ Verify that previous protocols with security vulnerabilities are included in the settings list</td></tr>
<tr><td>Security Threats</td><td>■ Older versions of protocols where vulnerabilities are discovered are likely to allow attackers to eavesdrop or manipulate data communications</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ ssl.enabled.protocols (default: TLSv1.2) : Set whether to request client authentication</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Keep the latest protocol set to TLSv1.2 or TLSv1.3 and without any vulnerabilities found</td></tr>
<tr><td>Vulnerable
■ The setting is empty or contains an unnecessary older version of the protocol.
■ State of using an unsafe protocol</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Maintain the latest security-secure protocols
■ Only TLSv1.2 is supported for Java 8 versions and later
■ For Java 11 and later versions, TLSv1.2, TLSv1.3 can be set
■ kafka/config/kraft/server.properties
ssl.enabled.protocols =TLSv1.3</td></tr>
</table>

<table>
<tr><th>No 5.3</th><th>Protocol mapping</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check the appropriateness of security protocol mapping by listener</td></tr>
<tr><td>Security Threats</td><td>■ Increased the likelihood of sensitive data leakage when low security protocols such as PLAINTEXT are used for internal traffic</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ listener.security.protocol.map : Define the mapping between the listener name and the security protocol</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Internal traffic is configured with encrypted protocols such as INTERNAL:SSL or INTERNAL:SASL_SSL
■ External traffic uses protocols with authentication and encryption enabled, such as EXTERNAL:SASL_SSL
■ Each listener's name is clearly defined and mapped to a protocol that matches the required security level.</td></tr>
<tr><td>Vulnerable
■ Set all listeners to non-encrypted protocols such as PLAINTEXT
■ Internal and external traffic is set to the same protocol on the same listener and no traffic is isolated
■ Unsuitable security levels are applied using the default values without explicit mapping.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ Encrypted protocol is recommended by default
■ Protect clusters from network attacks by separating internal and external traffic
■ Use SASL_SSL protocol if authentication is required
■ Add code to kafka/config/kraft/server.properties as in the following example
listener.security.protocol.map=INTERNAL:SSL,EXTERNAL:SASL_SSL,CONTROLLER:SSL</td></tr>
</table>

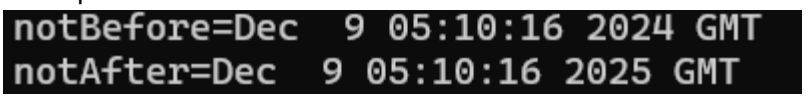
No 5.4	SSL Certificate Management
Overview	
Inspection Details	■ Periodically check the expiration date of the SSL certificate and verify that it is updated or replaced appropriately before expiration.
Security Threats	■ Secure communication between the cluster and the client becomes impossible when using expired certificates ■ Allows unauthorized users to intercept communications or use forged certificates to perform a man-in-the-middle attack
Inspection Targets and Evaluation Criteria	
Inspection Targets	■ broker, kafka/config/kraft/server.properties, SSL certificate validity
Evaluation Criteria	Good ■ Certificate expiration has not expired and the renewal cycle is being managed regularly ■ Certificate renewal process is clearly established
	Vulnerable ■ Expired certificate in use or near expiration of certificate ■ Status of possible service interruption after expiration due to unclear certificate renewal procedures
Recommended Actions	

Verifying Certificate Validation

■ Use the command below to verify the validity of the certificate

```
openssl x509 -in <certificate-file> -noout -dates
```

■ Example of Execution Photo



7. SASL (Simple Authentication and Security Layer)

7.1. Overview

Chapter 7 discusses SASL, an authentication system applicable to Kafka. SASL is a framework for performing authentication in network communication, providing various authentication mechanisms. Kafka can be SASL-applied to handle authentication and user authentication between clusters, thereby enhancing the security of the system. SASL supports a variety of authentication mechanisms, allowing you to choose a flexible authentication method that meets your system requirements.

This chapter will introduce each SASL mechanism, explain how to set it up, and which environment and which mechanisms are appropriate for users to choose. Through this chapter, we will understand the various SASL mechanisms applicable to Kafka and present how to apply the optimal settings for each environment.

7.2. SASL Configuration Guidelines

Below is an example of applying SASL to Kafka. This example is described based on PLAIN and recommends building a secure data transfer environment by applying it well to the system environment or considerations.

Step 1) Add broker settings

■ 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 : Definition of SASL to be used in protocol between broker internal communications
- Sasl.enabled.mechanisms : Mechanisms used with SASL

Step 2) Create kafka_server_jaas.conf at Broker

■ You can add users in the form of user_{username} = "{password}".

■ 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) Add Client Settings

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

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

Step 4) Create kafka_server_jaas.conf in Client

■ kafka/config/kafka_server_jaas.conf

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

Step 5) Add Kafka Server Shell Script Settings

■ kafka/bin/kafka-server-start.sh

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

Additional settings when using Zookeeper

If you are using Zookeeper instead of Kraft, the following additional settings are required.

■ kafka/config/zookeeper.properties

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

■ config/zookeeper_jaas.conf

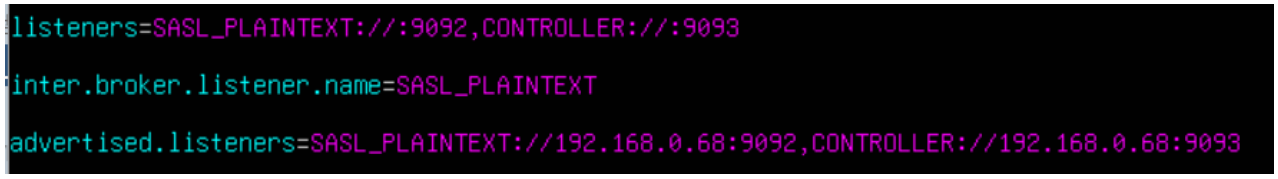
```
Server {
```

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

7.3. SASL Diagnostic Checklist

Classification	Sub-classification	Inspection Items
SASL	6.1	cryptographic communication
	6.2	Token Renewal
	6.3	SASL Authentication Message Size
	6.4	SASL login processing time and frequency

7.4. SASL Diagnostic Items

<table>
<tr><th>No 6.1</th><th>cryptographic communication</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check the use of encryption communication in SASL communication process</td></tr>
<tr><td>Security Threats</td><td>■ Increasing the likelihood of sensitive data leakage when SASL/PLAIN is used without encryption</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ listener (default: null): listener settings
■ advertised.listeners (default: null): Set the listener to be used during advertising
■ inter.broker.listener.name (default: null): Internal broker-to-broker listener settings</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ If SASL is PLAIN, the listener and inter.broker.listener.name are SASL_SSL or use SASL_PLAINTEXT and have mechanisms other than SASL/PLAIN</td></tr>
<tr><td>Vulnerable
■ Even though SASL is PLAIN, listener and inter.broker.listener.name are SASL_PLAINTEXT or use other methods other than SASL_PLAINTEXT and SASL_SSL.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Check kafka/config/kraft/server.properties configuration
■ Verify that the Listener, inter.broker.listener.name , advertised.listeners setting is set to SASL_SSL if SASL_PLAINTEXT or SASL/PLAIN
■ Write in kafka/config/kraft/server.properties as below example
 
</td></tr>
</table>

<table>
<tr><th>No 6.2</th><th>Token Renewal</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check that SASL login tokens are being updated in the right amount of time and in the right number of times</td></tr>
<tr><td>Security Threats</td><td>■ Risk that unnecessary update requests overpopulate authentication servers, leading to denial-of-service attacks as well as performance degradation
■ Too Long Token Renewal May Cause Frequent Service Disruption</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ sasl.login.refresh.buffer.seconds (default: 300): Defines the amount of time left to update the login token in seconds. Renew the token within this time.
■ ssl.login.refresh.min.period.seconds (default: 60): Defines the minimum interval, in seconds, between two consecutive update requests when updating a token.</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Status to process update requests with sufficient room before the update buffer time expires
■ The minimum interval between consecutive update requests is set to a value that prevents excessive load.</td></tr>
<tr><td>Vulnerable
■ Short refresh buffer time to load authentication server or potentially disrupt service with expired tokens
■ Excessive requests are sent to authentication servers beyond the interval, or the interval is too long, and there is a high possibility of renewal failure.</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Set to appropriate settings
■ sasl.login.refresh.Set buffer.seconds to a value greater than 60 seconds
■ Set sasl.login.refresh.min.period.seconds to a value greater than 30 seconds, less than 300 seconds
■ Additionally, monitor kafka cluster environments and certification service response times to flexibly adjust to the right time</td></tr>
</table>

<table>
<tr><th>No 6.3</th><th>SASL Authentication Message Size</th></tr>
<tr><th colspan="2">Overview</th></tr>
<tr><td>Inspection Details</td><td>■ SASL Message Size Appropriateness Check</td></tr>
<tr><td>Security Threats</td><td>■ Excessive size messages could lead to denial-of-service attacks by attackers
■ Too small size can also deny normal client requests, limiting service use</td></tr>
<tr><th colspan="2">Inspection Targets and Evaluation Criteria</th></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ ssl.server.max.receive.size (default: 1048576): Define the maximum size of requests a server can receive</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ Size limit settings are configured to match system performance, average message size, so that excessive messages do not cause resource exhaustion or normal message blocking</td></tr>
<tr><td>Vulnerable
■ Excessive messages quickly drain resources if the value is too large, allowing an attacker to exploit a denial-of-service attack
■ The value is too small to handle the client's normal authentication message.</td></tr>
<tr><th colspan="2">Recommended Actions</th></tr>
<tr><td colspan="2">Consideration
■ Check sasl.server.max.receive.size at kafka/config/kraft/server.properties
■ Client authentication messages are typically small, so it is appropriate to set 1 to 5 MB
■ Analysis of the size of authentication messages used by the service through real-time monitoring.Adjust the value of server.max.receive.size for your environment</td></tr>
</table>

<table>
<tr><th>No 6.4</th><th>SASL login processing time and frequency</th></tr>
<tr><td colspan="2">Overview</td></tr>
<tr><td>Inspection Details</td><td>■ Check if SASL login processing time and frequency is appropriate</td></tr>
<tr><td>Security Threats</td><td>■ A client might fail to establish a server connection, resulting in repeated authentication failures
■ Possibility of a denial-of-service attack with an excessive retry request by an attacker</td></tr>
<tr><td colspan="2">Inspection Targets and Evaluation Criteria</td></tr>
<tr><td>Inspection Targets</td><td>■ Broker, kafka/config/kraft/server.properties
■ sasl.login.connect.timeout.ms (default: null): Defines the maximum time when a connection is attempted to process a login request.
■ sasl.login.retry.backoff.ms (default : 300): Long wait between failed login requests and retries
■ sasl.login.retry.backoff.max.ms (default: 60000): Define maximum wait time between retries on login request failure</td></tr>
<tr><td rowspan="2">Evaluation Criteria</td><td>Good
■ The value of sasl.login.connect.timeout.ms allows the client to connect to the server normally.
■ A condition in which the retry interval is set appropriately in case of failure and does not overload the broker's accreditation service.</td></tr>
<tr><td>Vulnerable
■ sasl.login.connect.timeout.ms settings are too low to establish a connection or the server handles excessive requests
■ A condition that occurs repeatedly and excessively to the broker due to excessive retries
■ sasl.retry.backoff.max.ms is too large for client authentication delay and service impossible</td></tr>
<tr><td colspan="2">Recommended Actions</td></tr>
<tr><td colspan="2">Consideration
■ sasl.login.connect.timeout.ms = 5000
■ sasl.login.retry.backoff.ms = 500</td></tr>
</table>

- sasl.login.retry.backoff.max.ms = 10000
- The recommended values for each setting are as follows, and we recommend that you adjust the service environment and network status through monitoring.

Supplementary Provisions

<1. Minimum required per request>

Protocol (API Key)	Operation	Resource	Note
PRODUCE (0)	Write	TransactionalId	Permissions required for Transaction Producers with Transaction ID set
PRODUCE (0)	IdempotentWrite	Cluster	Requires applicable permission for demonstration of production
PRODUCE (0)	Write	Topic	For common producers, you need that privilege
FETCH (1)	ClusterAction	Cluster	Requires ClusterAction privileges for cluster resources to patch partition data
FETCH (1)	Read	Topic	For typical consumers, READ permission is required for each partition
LIST_OFFSETS (2)	Describe	Topic	
METADATA (3)	Describe	Topic	
METADATA (3)	Create	Cluster	Upon activation of auto-creation of topics, the broker checks permissions on cluster resources. Check permissions for Topic resources below if none exists
METADATA (3)	Create	Topic	Examine privileges on Topic resources if permissions do not exist for cluster resources when auto-creation of topics is enabled

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	Requires permission to Group and Topic
OFFSET_COMMIT (8)	Read	Topic	Requires permission to read operations as it is part of the consumption process
OFFSET_FETCH (9)	Describe	Group	Requires permission to Group and Topic
OFFSET_FETCH (9)	Describe	Topic	
FIND_COORDINATOR (10)	Describe	Group	If the request comes from Group
FIND_COORDINATOR (10)	Describe	TransactionalId	If Transaction Producer find Transaction Coordinator
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	Scan permissions to cluster resources first. Check Topic permissions for groups as shown below
LIST_GROUPS (16)	Describe	Group	Send empty responses if you do not have Group privileges.Applies from 2.1

SASL_HANDSHAKE (17)			
API_VERSIONS (18)			Communications that occur prior to authentication
CREATE_TOPICS (19)	Create	Cluster	Check permission priority for cluster resources
CREATE_TOPICS (19)	Create	Topic	For Topic, apply from 2.0
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 Resource Priority Scan
OFFSET_FOR_LEADER_EPOCH (23)	Describe	Topic	For Topic, apply from 2.1
ADD_PARTITIONS_TO_TXN (24)	Write	TransactionalId	Applies to Transaction requests only. Write permissions check for Topic after checking these resource permissions
ADD_PARTITIONS_TO_TXN (24)	Write	Topic	
ADD_OFFSETS_TO_TXN (25)	Write	TransactionalId	Applies to Transaction requests only. Read permissions check for groups after checking these resource permissions
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	Check Cluster Level Permissions on Broker Config Requests
DESCRIBE_CONFIGS (32)	DescribeConfigs	Topic	Check Topic Level Privileges on Topic Config Requests
ALTER_CONFIGS (33)	AlterConfigs	Cluster	Check Cluster Level Permissions for Broker Config Changes
ALTER_CONFIGS (33)	AlterConfigs	Topic	Check Topic Level Privileges When Changing Topic Configs
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)			Create a delegation token
CREATE_DELEGATION_TOKEN (38)	CreateTokens	User	Create a delegation token for user resources
RENEW_DELEGATION_TOKEN (39)			Renew delegation token
EXPIRE_DELEGATION_TOKEN (40)			Delegation Token Expires
DESCRIBE_DELEGATION_TOKEN (41)	Describe	DelegationToke	Delegated Token Overview

		n	
DESCRIBE_DELEGATION_TOKEN (41)	DescribeTokens	User	user resource delegation token describe
DELETE_GROUPS (42)	Delete	Group	
ELECT_PREFERRED_LEADERS (43)	ClusterAction	Cluster	
INCREMENTAL_ALTER_CONFIGS (44)	AlterConfigs	Cluster	Check Cluster Level Permissions for Broker Config Changes
INCREMENTAL_ALTER_CONFIGS (44)	AlterConfigs	Topic	Check Topic Level Privileges When Changing Topic Configs
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	