KIP-395: Encypt-then-MAC Delegation token metadata

This page is meant as a template for writing a KIP. To create a KIP choose Tools->Copy on this page and modify with your content and replace the heading with the next KIP number and a description of your issue. Replace anything in italics with your own description.

Status

Current state: *"Under Discussion"*

Discussion thread: *here [Change the link from the KIP proposal email archive to your own email thread]*

JIRA: *KAFKA-7691*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Currently Delegation token metadata (i.e. token information about "version", "tokenId", "owner", "renewers", "issueTimestamp", "maxTimestamp", "expiryTimestamp") is stored unencrypted in Zookeeper using JSON format. Apache Zookeeper 3.4 does not support SSL/TLS. In case of an insecure Zookeeper deployment, communication between Kafka brokers and a Zookeeper cluster might be intercepted and altered. This can lead to security incidents.

Until a stable Zookeeper release supports SSL/TLS, Kafka brokers could implement a strategy called Encrypt-then-MAC to encrypt sensitive metadata information about delegation tokens and protecting integrity using the master secret key (i.e. delegation.token.master.key) as a shared secret.

Proposed Changes

- Introduce a new broker configuration: **delegation.token.metadata.encryption.enable** to allow users to configure Kafka brokers to use delegation tokens with encrypted metadata. By default, delegation.token.metadata.encryption.enable shall be true.
- **Encrypt-then-MAC** will be implemented in the following way
  - Introduce a new utility class that is able to encrypt and decrypt byte arrays.
  - Store delegation token metadata in base64 encoded raw bytes in Zookeeper in the following format:

    [0-2] Three bytes of a known tag value, e.g. ‘D’,’T’,’E’

    [3] One byte of version number, 0x1 for now

    [4-19] Sixteen bytes of salt/IV for PBKDF2/AES-CBC

    [20-51] 32 bytes of HMAC

    [52-N] All remaining bytes are encrypted data
  - Add new methods to KafkaZkClient so that it can encrypt / decrypt delegation token metadata using the new utility class if broker is configured for delegation token encryption. Brokers should expect and be able to retrieve unencrypted delegation token metadata from Zookeeper (e.g. tokens might have been created by an older Kafka version).
  - Modify DelegationTokenManager so that it passes configured master key to KafkaZkClient if encryption is enabled.
  - Adjust existing tests and add new ones to cover new functionality.
- Document **upgrade strategy** (see details below)

Compatibility, Deprecation, and Migration Plan

- Kafka brokers will be able to read old, unencrypted delegation token metadata from Zookeeper.
- Users can decide whether Kafka brokers shall use delegation token metadata encryption.
  - If **delegation.token.metadata.encryption.enable** is false, there is no impact.
  - If **delegation.token.metadata.encryption.enable** is true, brokers store metadata of delegation tokens in an encrypted format together with message authentication code when a user creates or updates a token.
- If delegation token metadata encryption is enabled, a user shall follow a specific **upgrade strategy** to avoid a situation when brokers with different capabilities exist in the cluster:

- Before upgrading from Apache Kafka 1.0.0 to Apache Kafka 2.x.y, ensure that you set **delegation.token.metadata.encryption.enable** to **false**.
- Perform a rolling upgrade of brokers.
- After all brokers have been upgraded, set **delegation.token.metadata.encryption.enable** to **true**.
- Perform a rolling upgrade of brokers to enable delegation token metadata encryption.

This is a good practice because newer brokers might create delegation tokens with encrypted metadata that older brokers would not be able to use during the upgrade. If you need to rollback to the older version, and you have not set delegation.token.metadata.encryption.enable to false, brokers will not be able to parse encrypted metadata of new delegation tokens.

Alternatives

- Similar to the way SCRAM credentials are stored, it would be nice to allow users to store Delegation token metadata in external credential stores (for example Hadoop KMS). A plugin mechanism was not considered in the original design, and it might take significant effort to refactor existing implementation.
- Wait until a new release of Apache Zookeeper that supports SSL/TLS and update Kafka to use it. As of writing, GA of 3.6 is not known, and upgrading to a new Zookeeper version might be risky without comprehensive testing.
- Clients may decide whether to use encryption. It requires a new version of the token acquisition and renewal requests in the Kafka protocol.
- Introduce new znodes for encrypted delegation token metadata. It would introduce multiple source of truth.

Future work

- Add a new tool, **kafka-delegation-tokens-encrypt.sh** to encrypt all metadata in Zookeeper of all existing delegation tokens in the system.