KIP-152 - Improve diagnostics for SASL authentication failures

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: *KAFKA-4764*

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

Motivation

With the current Kafka SASL implementation, broker closes the client connection if SASL authentication fails without providing feedback to the client to indicate that authentication failed. Clients see this as a disconnection during authentication which may be related to authentication failure, but could also be due to broker failure. Client logs a warning with a hint that the disconnection was likely to have been due to an authentication failure, but does not handle this as a fatal security exception since the error may not be due to an authentication failure. Producers and consumers retry, attempting to create successful connections, treating authentication failures as transient failures.

Goals

1. Improve diagnostics for SASL authentication failures.
2. Reduce retries when authentication fails. Authentication failures should be treated as non-retriable exceptions rather than transient failures.
3. Reduce blocking in clients when authentication fails. If connection to a broker fails authentication, metadata wait operations in producers and consumers should avoid unnecessary blocking and throw an exception indicating authentication failure.

SSL Authentication Failures

This KIP does not change protocol-level handling of SSL authentication failures. Moving to a consistent protocol for SSL will break compatibility with existing brokers. But we have improved diagnostics for SSL by converting SSL exceptions to AuthenticationException under

⚠ Unable to render Jira issues macro, execution error.

. The system property `javax.net.debug` can be also set on the client to obtain comprehensive diagnostics for this case,

Public Interfaces

SASL does not provide a mechanism-independent way of reporting authentication failures. So this KIP proposes to add error reporting for SASL using the Kafka protocol. A new `SaslAuthenticate` request will be added to enable this. SASL authentication messages are currently length-encoded message blobs that are processed by the `SaslServer/SaslClient` implementations for the SASL mechanism. This KIP proposes to wrap these message blobs in `SaslAuthenticate` request/response messages defined in the Kafka protocol.

```
SaslAuthenticate Request (Version: 0) => sasl_auth_bytes
  sasl_auth_bytes => BYTES
SaslAuthenticate Response (Version: 0) => error_code sasl_auth_bytes
  error_code => INT16
  error_message => NULLABLE_STRING
```

```
sasl_auth_bytes => BYTES
```

`SaslHandshake` request version will be bumped up from `v0` to `v1`, without any change to the request or response format. For compatibility between 0.10 /0.11 clients and 1.0.0 brokers, the new `SaslAuthenticate` requests will be used only if `SaslHandshake v1` is used to initiate handshake.
A new error code `AUTHENTICATION_FAILED` will be added along with a corresponding `AuthenticationFailedException`.
1.0.0 Java clients will send an `ApiVersions` request prior to sending `SaslHandshake` request and use `SaslHandshake v1` only if supported by the broker. Otherwise, the older format authentication will be used, enabling 1.0.0 clients to authenticate with 0.10/0.11 brokers. 1.0.0 brokers will expect older format authentication unless `SaslHandshake v1` is used to initiate the handshake, maintaining compatibility with older clients.

Proposed Changes

SASL authentication sequence

- `ApiVersions` request is sent by the client to determine `SaslHandshake` version prior to authentication. This is supported by 0.10/ 0.11 brokers, retaining compatibility between 1.0.0 clients and older brokers. Since 0.10/0.11 brokers treat schema exceptions in the first request as GSSAPI auth message for compatibility with 0.9.x clients, `ApiVersions v0` request will be used. As no throttling is performed prior to authentication, `v0` is sufficient for the initial request.
- If `SaslHandshake` version is `v0`, then the older wire format is used for authentication. SASL authentication messages are sent as length-encoded byte arrays without wrapping the messages with Kafka protocol headers.
- If `SaslHandshake` version is `v1`, clients will send SASL authentication messages using the new `SaslAuthenticate` request/response format, where the actual SASL authentication message blobs are wrapped in the Kafka protocol. The network layer code for SASL already does creation and parsing of Kafka protocol requests and responses to process `SaslHandshake` request, so this will be a straightforward change.
- The error code in the final message from the broker will indicate if authentication succeeded or failed. In the case of authentication failure, a `SaslAuthenticate` response will be sent with `error_code` indicating authentication failure and empty `sasl_auth_bytes` before the connection is closed by the broker. Successful authentications will be logged at debug level. Failed authentications will be logged as warnings.

Authentication error handling

- If authentication fails, broker will return an error response and then close the connection.
- Java clients will log a warning for authentication failures, distinguishing these from EOF exceptions due to connection failure.
- Nodes to which connection failed due to authentication failure will be blacked out for the reconnect backoff interval.
- Producer *`waitForMetadata`* and consumer *`ensureCoordinatorReady`* will be updated to throw `AuthenticationFailedException` if connection to a broker fails authentication.

Versioning of SaslHandshake and SaslAuthenticate requests

- SaslHandshake version will be bumped up from 0 to 1 without any changes to the schema. SaslHandshake `v1` indicates that broker supports SaslAuthenticate requests.
- For future versions, SaslHandshake and SaslAuthenticate requests versions may be updated independently. ApiVersions request will be sent prior to SASL authentication to determine versions supported for SaslHandshake and SaslAuthenticate.
- SaslHandshake request version will be bumped up if there are any changes required to the protocol used to select SASL mechanisms.
- SaslAuthenticate request version will be bumped up if any additional field is added to the requests or responses, e.g. to support new error paths. Version may also be bumped up if new error codes are added which require special handling.
- SaslHandshake and SaslAuthenticate request versions will not be bumped up to enable new mechanisms unless the new mechanism requires schema/protocol changes to the existing SaslHandshake/SaslAuthenticate request/response format.

Compatibility, Deprecation, and Migration Plan

What impact (if any) will there be on existing users?

The wire-protocol changes from this KIP will be used only if `SaslAuthenticate` requests are supported by both the broker and the client. So existing clients and brokers will not be impacted.

0.9.x client => 1.0.0 broker

0.9.x clients start GSSAPI authentication straight after TLS handshake. 0.9.x clients connecting to 1.0.0 brokers will be handled by treating the first message as a GSSAPI auth message if it is not an `ApiVersions` or `SaslHandshake` request. This is already supported in the broker.

0.10.x, 0.11.0 client => 1.0.0 broker

0.10.x and 0.11.0 clients send `SaslHandshake v0` request as the first message. 1.0.0 brokers will use the current length-encoded SASL authentication message blobs that are not wrapped with Kafka request/response headers when v0 handshake request is used.

1.0.0 client => 0.10.x, 0.11.0 broker

1.0.0 clients connecting to 0.10/0.11 brokers will send an `ApiVersions` request to determine if `SaslHandshake v1` is supported. 0.10.x brokers already respond to versions request prior to SASL handshake. We will use `ApiVersions v0` for the initial request prior to handshake since no throttling is performed prior to authentication. 1.0.0 clients will use `SaslHandshake v0` to authenticate with 0.10.x and 0.11.0 brokers using the current length-encoded SASL authentication messages that are not wrapped with Kafka request/response headers.

If we are changing behavior how will we phase out the older behavior?

We will continue to support the older behavior. There is no plan to remove support for the current `SaslHandshake v0` request combined with the current length-encoded authentication message blobs at the moment. We will also continue to support the older 0.9.x GSSAPI clients which don't send handshake requests.

Rejected Alternatives

Send an error code if SASL authentication fails

We could retain the existing wire protocol for SASL authentication and add an authentication error code that is sent by the broker if SASL authentication fails, just before closing the connection. This will be treated as an invalid token by the SASL client authenticator, and the error handling for invalid tokens can be updated to report authentication failure for this case. This is a bit of a hack, but would work with GSSAPI, PLAIN and SCRAM. The PR for KAFKA-4764 currently implements this and has been tested with all the mechanisms in Kafka. The current proposal in this KIP is a bigger change, but is future-proof.

Add just an error code to SASL authentication messages from the broker without adding the entire Kafka request/response header.

Most of the fields in the headers like client-id are irrelevant for SASL messages which are treated as a blob by Kafka. Hence it may be unnecessary to add the entire header. Just a status/error code would be sufficient. Kafka request format was chosen since we already use this format for `SaslHandshake` requests, so the authenticators have knowledge of Kafka requests. Also, this enables the API to evolve in future if required.

Try all brokers before throwing AuthenticationException in clients to handle the case where credentials are being updated

It will be difficult to handle the case where credentials are different on brokers during an update since connection to one broker to obtain metadata may succeed while the subsequent produce to another broker could fail. It may be possible to improve the behaviour in future within the authentication handler by refreshing credentials when authentication fails. For now, authentication exception will be thrown on first failure so that misconfigured applications fail fast.