KIP-85: Dynamic JAAS configuration for Kafka clients

Status

Current state: *"Accepted"*

Discussion thread: *here*

JIRA: *KAFKA-4259*

Released: 0.10.2.0

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

Motivation

Kafka currently uses JAAS login configuration files for SASL authentication. The login configuration can be supplied in one of the following ways:

1. System property `java.security.auth.login.config`
2. Java security properties file (`lib/security/java.security` of the JRE), property `login.config.url.n`
3. Programmatically by invoking `javax.security.auth.login.Configuration.setConfiguration(Configuration)`

1) is the most commonly used method (and the documented way) of configuring login context in Kafka. This requires access to a file containing the login configuration. Many applications which obtain credentials during runtime write a temporary file to use this approach (eg. https://developer.ibm.com/bluemix/2016/02/19/managing-iot-devices-with-kafka-and-mqtt/)
2) is typically not used by client applications since it requires write access to the JRE installation and modifies the configuration for all applications using the JRE.
3) is the only programmatic interface to set Configuration. This is a JVM-wide setting and is typically the cause of many bugs, including in Kafka tests. Programmatic creation of `Configuration` objects is not very straightforward when credentials are known only at runtime. Also, the caching of JVM-wide `Configuration` instance in Java makes it difficult to identify the source of bugs when configuration is updated dynamically.

To enable cloud applications to use Kafka without relying on a physical file to store credentials (eg. CloudFoundry applications without persistent storage), it will be useful to add a configuration option that simplifies JAAS configuration for Kafka clients. This will also be useful to configure multiple `KafkaClient` login contexts when multiple users are supported within a JVM (KIP-83).

Public Interfaces

Configuration property

Name: `sasl.jaas.config`

Type: `PASSWORD` (to prevent credentials from being logged eg. SASL/PLAIN password)

Format: The same format as a login context entry in JAAS configuration:

```
<LoginModuleClass> <ControlFlag> *(<OptionName>=<OptionValue>);

ControlFlag = required / requisite / sufficient / optional
```

Example:

```
org.apache.kafka.common.security.plain.PlainLoginModule required username=”alice” password=”alice-secret”;
```

This property will be enabled only in Kafka clients. Even though brokers can also use the same approach for setting JAAS configuration dynamically, there is currently no requirement to support dynamic JAAS configuration for brokers. This avoids the need to implement multiple login module support in `sasl.jaas.config`. Broker-side implementation can easily be added in future if required.

`sasl.jaas.config` will be of type `PASSWORD` to avoid exposing credentials in logs. As with the current JAAS configuration files, this has the drawback that configuration issues cannot be debugged using log files alone. But this format was chosen to enable extensible mechanism-independent configuration that is consistent with the existing JAAS configuration.

Post Implementation Note: This config was added to brokers in 1.1.0 under KIP-226 - Dynamic Broker Configuration

Proposed Changes

Default JAAS configuration

If `sasl.jaas.config` is not specified, the JAAS configuration instance configured in the JVM using any of the three methods described in the Motivation section is used. The implementation will return `Configuration.getConfiguration()` that caches the configuration instance that is loaded. This is the current behavior.

JAAS configuration using client properties

If `sasl.jaas.config` is specified, a new `Configuration` instance is created with a single entry corresponding to the login type (`KafkaClient` for clients). The generated `Configuration` instance will not be set as the JVM-wide static value. Login context will be created using the constructor that accepts a configuration object rather than use the static value `Configuration.getConfiguration()`.

```
LoginContext(String name, Subject subject, CallbackHandler callbackHandler, Configuration config)
```

When `LoginManager` caching in Kafka is updated to support multiple users in a JVM (KIP-83 is addressing multiple users), `sasl.jaas.config` can be set to different values for different clients to enable multiple users without manipulating JVM-wide `Configuration` instances or adding additional mechanism-specific properties to identify users. All SASL mechanisms will be handled consistently using this approach.

Compatibility, Deprecation, and Migration Plan

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

  None

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

  Existing behaviour will be retained as default.

Test Plan

Since most of the code path will be tested by existing tests, no additional system tests are required. Unit and integration tests will be added to ensure that JAAS configuration can be set and modified programmatically without changing JVM-wide values.

Rejected Alternatives

Add a configurable public interface for JaasConfigProvider

Rather than a property that enables the creation of a `javax.security.auth.login.Configuration` instance, it would be sufficient to expose a configuration provider, leaving it up to applications how the configuration instance is created. This simplifies the Kafka implementation, but makes it harder for applications to generate configurations. Additional properties will be required to configure the provider.

Add separate properties for login module options instead of a composite property using jaas.conf format

The KIP proposes to use a single property `sasl.jaas.config` containing all the options of a login context. Separate properties (eg. sasl.jaas,login.context, sasl.jaas.username, sasl.jaas.password etc.) may make it easier to parse the configuration. But since login option names are dependent on the login module class and the SASL mechanism, it is more flexible to use a single Kafka property in the standard JAAS format that captures the contents of a login context.