KIP-127: Pluggable JAAS LoginModule configuration for SSL

Status

Current state: *Rejected*

Discussion thread: *here*

JIRA: *KAFKA-4784*

Released: <Kafka Version>

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

Motivation

Kafka currently supports providing a JAAS LoginModule for pluggable authentication when using SASL and SASL_SSL. When using the SASL_SSL channel, the channel will override the setting for requiring a client certificate to be presented so it is not possible to require client authentication by providing a certificate. This makes sense for SASL_SSL as the SASL mechanism will be used for authentication instead of a client certificate in this case. However, this means that the SSL channel needs to be used when a client certificate is required for authentication. Currently when using SSL the only authentication that is done is the SSL handshake between client and server. It would also be ideal to be able to provide an option for custom authentication for the SSL channel that uses the client's X509 credentials that goes beyond the SSL handshake. This would allow providing extra authentication based on a user's custom requirements.

Public Interfaces

Configuration changes

An optional JAAS LoginModule for the SSL channel can be configured using the namespace SslKafkaServer. The implementation will be provided by the end user on the classpath. For example:

```
SslKafkaServer {
    com.some.login.module.UserSslLoginModule required
    option="value"
    option2="value";
};
```

`-Djava.security.auth.login.config` will still be used to specify the location of the configuration file.

API change

An abstract X509LoginModule class can be provided to make it easier for a user to custom build a LoginModule.

A new `X509Callback` class and `X509CallbackHandler` class in the package org/apache/kafka/common/security/authenticator will allow a user's JAAS LoginModule implementation to get access to the X509 credentials. For example:

```
public class TestSslLoginModule implements LoginModule {
    @Override
    public void initialize(Subject subject, CallbackHandler callbackHandler, Map<String, ?> sharedState,
            Map<String, ?> options) {
        Callback[] callbacks = new Callback[1];
        X509Callback callback = new X509Callback();
        callbacks[0] = callback;
        try {
            callbackHandler.handle(callbacks);
        } catch (Exception e) {
            //handle exception
        }
                //acquire X509 credentials
        Certificate[] certs = callback.getPeerCertificates();
        X509Certificate x509 = (X509Certificate) certs[0];

    }

    @Override
    public boolean login() throws LoginException {
        //do custom login logic
        return true;
    }
    //Rest of methods below
}
```

Proposed Changes

New JaasContext.Type called SSL_SERVER and new Global Context name called `SslKafkaServer`. A user will be able to provide a custom JAAS module under the name of `SslKafkaServer`. This is in addition to the existing ways to configure a SASL server with KafkaServer. By using `SslKafkaServer` a separate LoginModule configuration can be provided for both SSL and SASL/SASL_SSL.

When the `ChannelBuilders` class builds an SSL channel it will try and load a matching JAAS context. If the context exists it will be provided to the SslChannelBuilder class. If it can't be loaded then it will not be used.

A new `SslServerAuthenticator` will be created and will use the JAAS module to authenticate. When authenticate() is called the LoginManager will be loaded and authentication will be done if the JAAS context has been provided. This needs to be done here in order to make sure the SSL channel has already finished the handshake.

`SsTransportLayer` will have a new method called getPeerCertificates() to get the peer X509 credentials.

The principal will be extracted from the Subject provided by the JAAS LoginModule. If the LoginModule does not set a principal in the Subject then the `SslServerAuthenticator` will use the configured `PrincipalBuilder` object to build the Principal.

Compatibility, Deprecation, and Migration Plan

There will be no impact on existing users if they do not configure a JAAS module for the SSL channel. This is an optional configuration and everything should continue to work as it did before. The `PrincipalBuilder` class will still be used to build the principal if there is no configured JAAS module. If a JAAS module is configured then the `SslServerAuthenticator` will first try and use a principal set on the Subject from the JAAS module but will fallback to the `PrincipalBuilder` class if one is not set which should preserve backwards compatibility.

Test Plan

There will be unit tests written based off of SslTransportLayerTest for the new configuration and the existing tests that will run for both unit and system tests will show that nothing has been broken for existing users.

Rejected Alternatives

Custom Authenticator

A change could be made to allow a user to configure a custom Authenticator but this would not be a standard API and would be a Kafka only API implementation. It is better to allow the configuration of a JAAS LoginModule as that is an existing standard and is more flexible.

Provide a X509CallbackHandler for SASL_SSL

The SASL_SSL channel disables client authentication so a certificate can't be required. Also using X509 credentials for SASL_SSL doesn't make sense as it doesn't fit into the SASL protocol.