KIP-684 - Support mutual TLS authentication on SASL_SSL listeners

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: ⚠ Unable to render Jira issues macro, execution error.

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

Motivation

SSL listeners in Kafka can configure TLS client authentication (also known as mutual TLS authentication) using the broker configuration option `ssl.client.auth` with the following values:

- `none` (default) : disables client authentication, assigns `User:ANONYMOUS` as `KafkaPrincipal`.
- `requested` : enables optional client authentication, uses distinguished name (DN) from certificate as `KafkaPrincipal` by default if certificate provided, `User:ANONYMOUS` otherwise.
- `required` : enables mandatory client authentication, uses DN from certificate as `KafkaPrincipal` by default.

Kafka currently disables TLS client authentication for SASL_SSL listeners even if `ssl.client.auth` is configured. This behaviour was introduced at a time when this configuration option could only be configured broker-wide. So in a broker with one SSL listener and one SASL_SSL listener, `ssl.client.auth=required` would have forced both listeners to support TLS client authentication. In the common case where internal listeners used TLS client authentication and SASL_SSL used SASL authentication without requiring key store distribution, this behaviour was not desirable.

KIP-103 introduced listener names and listener-prefixed configuration options, enabling security options to be configured at individual listener level. But we continue to ignore `ssl.client.auth` configuration for SASL_SSL listeners. This KIP proposes to support TLS client authentication for SASL_SSL listeners for additional protection in security-critical deployments. In organizations where mutual TLS authentication is mandatory on all connections, this feature enables TLS authentication to be combined with SASL-based client identity.

Goals

1. Support mutual TLS authentication (mTLS) for SASL_SSL listeners to increase security of SASL_SSL and satisfy mandatory controls in security-critical deployments.
2. Since distribution of TLS certificates with client identity as DN adds significant certificate management overhead, many organizations are unable to use mTLS in external listeners with SSL as the security protocol. Support mTLS in SASL_SSL listeners to enable the use of shared client key stores with SASL-based client identity.
3. Retain current defaults that disable mTLS for both SSL and SASL_SSL unless explicitly configured.
4. Retain current semantics for the broker-wide `ssl.client.auth` configuration for compatibility.
5. Enable mTLS for SASL_SSL if listener-level `ssl.client.auth` configuration option is set to `required` or `requested`, supporting both mandatory and optional TLS client authentication.
6. Retain current `KafkaPrincipal` based on SASL authentication for SASL_SSL connections even if mTLS is enabled, avoiding any impact on authorization or quotas.

Public Interfaces

Configuration Options

No new configuration options are being added, but we will add support for listener-prefixed `` `ssl.client.auth` `` for SASL_SSL listeners. Broker-wide `` `ssl.client.auth` `` without listener prefix will apply only to SSL listeners as it is today.

`` `ssl.client.auth` `` will continue to accept the following three values:

- `` `none` `` (default): Client certificates will not be requested by brokers.
- `` `requested` `` : Client certificates are optional may be provided by clients. If provided, clients will fail authentication if certificates are not valid.
- `` `required` ``: Client certificates are mandatory and must be provided. Clients will fail authentication if certificates are not valid.

SSL listeners will continue to use this configuration with the following order of precedence (*no change*):

1. `listener.name.<sslListenerName>.ssl.client.auth`
2. `ssl.client.auth`

SASL listeners will only support this configuration with the listener prefix (*new behaviour*):

1. `listener.name.<saslListenerName>.ssl.client.auth`

If `` `ssl.client.auth` `` is configured without listener prefix and the broker has one or more SASL_SSL listeners, a warning will be logged to indicate that the option is applied only to SSL listeners and that the behaviour may change in future releases. We can consider removing the inconsistency for this configuration option and apply the value to all listeners in a future major release, either 4.0 or later.

Client Identity

SASL_SSL listeners will continue to use the client identity (`KafkaPrincipal`) established using SASL authentication regardless of whether mTLS is enabled. TLS client authentication is expected to be used for increased security of the connection, but not as a replacement for client identity. Authorization and quotas will continue to use the `KafkaPrincipal` established using SASL by default. Custom principal builders may use additional information from the SSL session for the `KafkaPrincipal` associated with SASL_SSL connections. Custom authorizers and custom quota callbacks may use additional information provided by custom principal builders for authorization decisions and quotas.

Public API Changes

`SaslAuthenticationContext` will be extended to return `SSLSession` for SASL_SSL listeners. This will be included regardless of whether mTLS is enabled. Custom principal builders may use this information to create principals that encapsulate SSL authentication state. For example, custom authorizers may restrict some operations to users who authenticated successfully using TLS certificates as well as SASL.

SaslAuthenticationContext

```
public class SaslAuthenticationContext implements AuthenticationContext {
    ....
    private final Optional<SSLSession> sslSession;

    public SaslAuthenticationContext(SaslServer server, SecurityProtocol securityProtocol, InetAddress
clientAddress, String listenerName) {
        this(server, securityProtocol, clientAddress, listenerName, Optional.empty());
    }
    public SaslAuthenticationContext(SaslServer server,
                                     SecurityProtocol securityProtocol,
                                     InetAddress clientAddress,
                                     String listenerName,
                                     Optional<SSLSession> sslSession) {
        this.server = server;
        this.securityProtocol = securityProtocol;
        this.clientAddress = clientAddress;
        this.listenerName = listenerName;
        this.sslSession = sslSession;
    }

    public Optional<SSLSession> sslSession() {
        return sslSession;
    }
    ....
}
```

Proposed Changes

`ChannelBuilders` will be updated to propagate the listener-prefixed value of `` `ssl.client.auth` `` to `SaslChannelBuilder`. At the moment `SaslChannelBuilder` creates `SslFactory` with `` `clientAuthConfigOverride=none` ``. That will be updated to use listener-prefixed configuration of `` `ssl.client.auth` `` with default value `` `none` ``.

`SaslServerAuthenticator` will be updated to create `SaslAuthenticationContext` with `SSLSession` if the transport layer is `SslTransportLaye`
.

No changes are expected on Java clients since we already allow configuration of SSL key stores with SASL_SSL that are used if broker requests certificates.

Compatibility, Deprecation, and Migration Plan

By default, `` `ssl.client.auth` `` will be `` `none` ``, retaining current behaviour. Also, by default, broker-wide `` `ssl.client.auth` `` configuration will not be applied to SASL_SSL listeners, avoiding issues during upgrade in brokers with a combination of SSL and SASL_SSL listeners.

Brokers will start enforcing listener-prefixed `` `ssl.client.auth` `` for SASL_SSL listeners. This is a breaking change for brokers that have been wrongly configured with `` `listener.name.<saslListenerName>.ssl.client.auth` `` to `required|requested`, expecting the configuration value to be ignored. We will document this in upgrade notes to ensure that users can remove the configuration before upgrading.

If broker-wide `` `ssl.client.auth` `` is configured in a broker with both SSL and SASL_SSL listeners, a warning will be logged to indicate that the option is applied only to the SSL listeners and that this behaviour may change in future releases. We can consider removing the inconsistency in the semantics of this configuration and apply the value to all listeners in a future major release, either 4.0 or later.

Test Plan

SASL_SSL with mutual TLS authentication will be tested using unit and integration tests.

Rejected Alternatives

Use a different configuration option for enabling mTLS with SASL_SSL

We could introduce a new configuration `sasl.ssl.client.auth` instead of using `ssl.client.auth` to avoid any change to the existing option. But overall, this seems more confusing since all other SSL options are shared between SSL and SASL_SSL.

Support broker-wide and listener-prefixed `ssl.client.auth` for SASL_SSL

All other SSL configs are supported at both levels for SSL and SASL_SSL. It may be confusing for just `ssl.client.auth` to require listener-prefix for SASL_SSL. But if broker-wide `ssl.client.auth` is enforced for SASL_SSL, it would be a breaking change that requires existing brokers that currently use this for one of the SSL listeners to change that to listener-prefixed config to avoid affecting SASL_SSL listeners. The current proposal limits impact on existing deployments except in the erroneous/unintentional setting of `listener.name.<saslListenerName>.ssl.client.auth`. We have other configs like `sasl.jaas.config` for SASL listeners that can only be set at listener/saslMechanism level, so this feels like a reasonable limitation for SASL listeners.

Support multiple KafkaPrincipals associated with single connection

The KIP proposes to use mTLS for additional protection of SASL_SSL listeners, with SASL identity as the default principal. This is different from ZooKeeper which associates multiple principals for SASL and SSL with the same connection, granting access when either principal has access. Since Kafka authorizers and quotas which rely on client identity work with a single identity, it would add more complexity and compatibility issues with multiple identities. So we will continue to associate a single client identity with each connection. Custom principal builders may combine SSL and SASL authentication information to determine this identity. Custom authorizers and custom quota callbacks can use the information from custom principals to determine the access control model and quotas when both TLS client authentication and SASL authentication were applied.