KIP-880: X509 SAN based SPIFFE URI ACL within mTLS Client Certificates

Status

Current state: *Under discussion*

Discussion thread: *KIP-880: X509 SAN based SPIFFE URI ACL within mTLS Client Certificates*

JIRA: *KAFKA-14340*

Motivation

Today there are several integration scenario's between Kakfa and Kubernetes and Istio hosted micro-services.

- Some prefer to deploy both consumers and producers, and brokers into Kubernetes and leverage Istio to take care of the mTLS.
- Some prefer to only deploy consumers and producers inside Kubernetes and deploy Kafka outside the K8s cluster (VM/BareMetal) or consume it as a SaaS offering.

This feature request is aimed at deployment scenario's where consumers and producers are deployed as micro-service within K8s and the broker is outside of the K8s cluster. As a Kafka consumer and producer, we want to be able to leverage Istio's build-in Client ID system, which is based on X509 SPIFFE URI's in the SAN extension field. Today, only the CN field can be leveraged as an AuthNZ mechanism in Kafka.

The mayor advantage of this proposal would be that we do not need to leverage PLAINTEXT or tunnel an other AuthNZ mechanism, but can directly use Istio/SPIFFE provided Client Identity with the mTLS based AutNZ and ACL rule mechanism, providing smoother integration, better security and increased performance.

Some external documentation references on the concept of use a SPIFFE ID as secure workload Identity:

- https://istio.io/latest/docs/concepts/security/#principals
- https://istio.io/v1.3/docs/concepts/security/#istio-security-vs-spiffe (Istio and SPIFFE share the same identity document: **SVID (SPIFFE Verifiable Identity Document)**. For example, in Kubernetes, the X. 509 certificate has the URI field in the format of spiffe://\<domain\>/ns/\<namespace\>/sa/\<serviceaccount\> .)
- https://spiffe.io/docs/latest/spiffe-about/spiffe-concepts/#spiffe-id

Public Interfaces

This is not a new interface, but an implementation of the ***KafkaPrincipalBuilder*** interface to add support for SPIFFE based SAN URIs and return them as a Principle so they can be leveraged to create ACL rules directly.

Proposed Changes

This is not a new interface, but an implementation of the ***KafkaPrincipalBuilder*** interface to add support for SPIFFE based SAN URIs and return them as a Principle so they can be leveraged to create ACL rules directly.

There are several POC implementations out there implementing a bespoke ***KafkaPrincipalBuilder*** implementation for this purpose. Two examples include

- https://github.com/traiana/kafka-spiffe-principal
- https://github.com/boeboe/kafka-istio-principal-builder (written by myself)

I can use some help here to determine the best implementation and improve the code in terms of resiliency and logging.

The proposal would be to include this functionality within Kafka's main functionality, so end-users do not need to bother with loading custom classes and leverage a vetted implementation instead.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*
- *If we are changing behavior how will we phase out the older behavior?*
- *If we need special migration tools, describe them here.*
- *When will we remove the existing behavior?*

Test Plan

Describe in few sentences how the KIP will be tested. We are mostly interested in system tests (since unit-tests are specific to implementation details). How will we know that the implementation works as expected? How will we know nothing broke?

Rejected Alternatives

If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.