TokenAuth Breakdown JIRAs

This breaks down the TokenAuth work discussed in HADOOP-9392 and the design, and based on this breakdown we’re going to create subtask JIRAs for better discussion, collaboration, and contribution. After all the main items are defined this doc will be updated then and complete.

Overview
As can be seen in the following figure, the whole TokenAuth work is divided into 3 parts: TokenAuth Framework, TokenAuth implementation – HAS, and TokenAuth Integration.

TokenAuth Framework

This defines Pluggable TokenAuth framework and common facilities as a main JIRA (subtask of HADOOP-9392) with all the essential elements as subtasks of the main JIRA.

HADOOP-9796 Pluggable TokenAuth framework and core facilities
As discussed in HADOOP-9392, we’re proposing a pluggable TokenAuth framework to abstract and address the requirements, goals and collaboration concerns already widely discussed in the JIRA with the design doc, and in community. In this JIRA, we'll:

- Define the framework itself, and clarifies the key goals, properties, and facilities that this framework should meet with and provide. Most of the points have already been explained in HADOOP-9392 and the TokenAuth design doc. To collaborate with HSSO and more importantly to allow other solutions, TokenAuth itself is just defined as a framework with required APIs, protocols, flows, and facilities along with some simple implementations for related constructs, entities and even services. The framework is neutral, no vendor specific, and subject to be widely discussed and defined together as a common effort of community. As the most important key point, the framework should be pluggable in all the key places to allow certain solutions to employ their own product level implementations. Based on this framework, Rhino will come up HAS solution. The framework related discussions in high level aspects can be in this separate umbrella JIRA, and sub task JIRAs will be opened to address each aspect of the framework.
- Define APIs for all the important entities and parties involved in TokenAuth framework.
- Define important procedures and protocols, for example, the protocol between token authn client and server.
• Implement this framework with the defined APIs, procedures and protocols. Meanwhile, leave pluggable extension points in key places for solutions to customize and implement with their own complicated mechanisms.

• Initially, we have the following items for the framework. It’s to be complemented. Each of the items will be defined and discussed separately in corresponding subtask JIRA.
  ✓ Token definition and API;
  ✓ TokenAuthn method for Hadoop RPC;
  ✓ Authentication Service API;
  ✓ Identity Token Service API;
  ✓ Access Token Service API;
  ✓ Attribute Service API;
  ✓ Token authentication client;
  ✓ Token cache for TokenAuth;
  ✓ Common configuration for TokenAuth;
  ✓ Hadoop token command;
  ✓ Key Provider API;
  ✓ Web SSO for TokenAuth;
  ✓ REST SSO for TokenAuth;
  ✓ Auditing for TokenAuth;
  ✓ And etc.

**TokenAuth Implementation**

This defines *TokenAuth Implementation – HAS* as a main JIRA (subtask of HADOOP-9392) with all the essential elements as subtasks of the main JIRA.

**HADOOP-9798 TokenAuth Implementation - HAS**

HAS is a complete and enterprise ready security solution based on TokenAuth framework proposed by HADOOP-9392 and utilizing the common facilities provided by the framework. It provides all the necessary implementations of entities, interfaces and services defined in the framework that’s required by industrial deployment.

As a major goal for Rhino, HAS addresses AAA (Authentication, Authorization and Auditing) concerns for Hadoop across the ecosystem. The 'A' of HAS could be explained as "Authentication", "Authorization", or "Auditing", depending on which role(s) HAS is configured with. In high level considerations, we may need Authentication Server, Authorization Server, or Auditing Server, and such servers would be great to be combined into one centralized server, or be deployed separately regarding performance or network concerns. Currently we’re mainly focusing on "Authentication" and "Authorization", and these two roles can be configured in one server instance or in separate server instances.

A more detailed scope of HAS implementation is as follows:

• Define and implement the common and management facilities shared across the implementation of different services. These include configuration mechanism for
services, persistent API and method for loading and storing data, auditing and logging API, shared high availability approach, REST API framework and authentication and so on.

- Define and implement Authentication Server role for HAS. The authentication server provides identity authentication service and issues identity token. The authentication can be configured with a chain of authentication modules for providing multi-factor authentication ability. By default, we will support AD (as LDAP) / LDAP authentication module and AD (as Kerberos) / Kerberos authentication module.

- Define and implement Authorization Server role for HAS. The authorization server includes service level authorization, access token issue and fine-grained authorization service.

- Implement Attribute Service for HAS, to allow integration of third party attribute authorities. The Attribute Service provides the ability to connect and retrieve attributes from different attribute sources such as LDAP or Database.

- Provides authorization enforcement library for Hadoop services to enforce security policies utilizing related services provided by the Authorization Server. To enforce the fine-grained authorization policies, the policies must be loaded, synchronized, and evaluated at Hadoop side.

**TokenAuth Integration**

This defines *TokenAuth Integration* as a main JIRA (subtask of HADOOP-9392) with all the essential elements as subtasks of the main JIRA.

**HADOOP-9799 TokenAuth Integration**

TokenAuth Integration includes tasks that employ TokenAuth framework and relevant solutions to enable related supports for various Hadoop components for typical enterprise deployments. Currently we have the following in mind:

- Enable Web SSO flow for web interfaces like HDFS and YARN;

- Enable REST SSO flow for REST interface like Oozie;

- Add Thrift and Hive JDBC support using TokenAuth. We consider this support because it is an important interface for enterprise to interact with data;

- And etc.