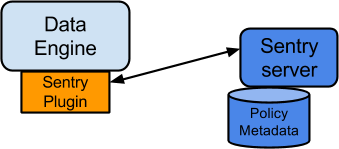
# Authorization With Apache Sentry

Apache Sentry is a granular, role-based authorization module for Hadoop. Sentry provides the ability to control and enforce precise levels of privileges on data for authenticated users and applications on a Hadoop cluster. Sentry currently works out of the box with Apache Hive, Hive Metastore/HCatalog, Apache Solr, Impala, and HDFS (limited to Hive table data).

Sentry is designed to be a pluggable authorization engine for Hadoop components. It allows you to define authorization rules to validate a user or application’s access requests for Hadoop resources. Sentry is highly modular and can support authorization for a wide variety of data models in Hadoop.

**Sentry Components**



There are three components involved in the authorization process:

* **Sentry Server**

The Sentry RPC server manages the authorization metadata. It supports interfaces to securely retrieve and manipulate the metadata.

* **Data Engine**

This is a data processing application such as Hive or Impala that needs to authorize access to data or metadata resources. The data engine loads the Sentry plugin and all client requests for accessing resources are intercepted and routed to the Sentry plugin for validation.

* **Sentry Plugin**

The Sentry plugin runs in the data engine. It offers interfaces to manipulate authorization metadata stored in the Sentry server, and includes the authorization policy engine that evaluates access requests using the authorization metadata retrieved from the server.

**Key Concepts**

* Authentication - Verifying credentials to reliably identify a user
* Authorization - Limiting the user’s access to a given resource
* User - Individual identified by underlying authentication system
* Group - A set of users, maintained by the authentication system
* Privilege - An instruction or rule that allows access to an object
* Role - A set of privileges; a template to combine multiple access rules
* Authorization models - Defines the objects to be subject to authorization rules and the granularity of actions allowed. For example, in the SQL model, the objects can be databases or tables, and the actions are SELECT, INSERT, CREATE and so on. For the Search model, the objects are indexes, collections and documents; the access modes are query, update and so on.

**User Identity and Group Mapping**

Sentry relies on underlying authentication systems, such as Kerberos or LDAP, to identify the user. It also uses the group mapping mechanism configured in Hadoop to ensure that Sentry sees the same group mapping as other components of the Hadoop ecosystem.

Consider a sample organization with users Alice and Bob who belong to an Active Directory (AD) group called finance-department. Bob also belongs to a group called finance-managers. In Sentry, you first create roles and then grant privileges to these roles. For example, you can create a role called Analyst and grantSELECT on tables Customer and Sales to this role.

The next step is to join these authentication entities (users and groups) to authorization entities (roles). This can be done by granting the Analyst role to the finance-department group. Now Bob and Alice who are members of the finance-department group get SELECT privilege to the Customer and Sales tables.

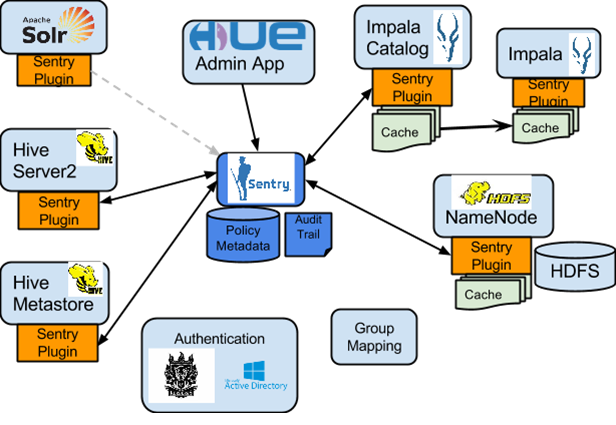
**Role-Based Access Control**

Role-based access control (RBAC) is a powerful mechanism to manage authorization for a large set of users and data objects in a typical enterprise. New data objects get added or removed, users join, move, or leave organisations all the time. RBAC makes managing this a lot easier. Hence, as an extension of the sample organization discussed previously, if a new employee Carol joins the Finance Department, all you need to do is add her to the finance-department group in AD. This will give Carol access to data from the Sales and Customer tables.

**Unified Authorization**

Another important aspect of Sentry is the unified authorization. The access control rules once defined, work across multiple data access tools. For example, being granted the Analyst role in the previous example will allow Bob, Alice, and others in the finance-department group to access table data from SQL engines such as Hive and Impala, as well as using MapReduce, Pig applications or metadata access using HCatalog.

**Sentry Integration with the Hadoop Ecosystem**



As illustrated above, Apache Sentry works with multiple Hadoop components. At the heart you have the Sentry Server which stores authorization metadata and provides APIs for tools to retrieve and modify this metadata securely.

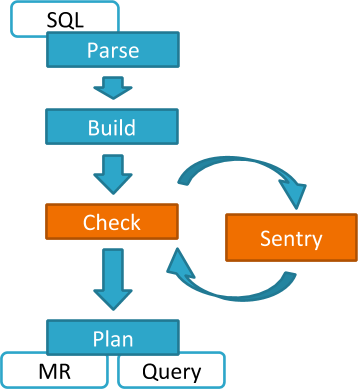
Note that the Sentry server only facilitates the metadata. The actual authorization decision is made by a policy engine which runs in data processing applications such as Hive or Impala. Each component loads the Sentry plugin which includes the service client for dealing with the Sentry service and the policy engine to validate the authorization request.

**Hive and Sentry**

Consider an example where Hive gets a request to access an object in a certain mode by a client. If Bob submits the following Hive query:

select \* from production.sales

Hive will identify that user Bob is requesting SELECT access to the Sales table. At this point Hive will ask the Sentry plugin to validate Bob’s access request. The plugin will retrieve Bob’s privileges related to the Sales table and the policy engine will determine if the request is valid.



Hive works with both, the Sentry service and policy files. Cloudera recommends you use the Sentry service which makes it easier to manage user privileges. For more details and instructions, see [The Sentry Service](https://www.cloudera.com/documentation/enterprise/5-7-x/topics/cm_sg_sentry_service.html#xd_583c10bfdbd326ba--6eed2fb8-14349d04bee--76f5)*or* [Sentry Policy File Authorization](https://www.cloudera.com/documentation/enterprise/5-7-x/topics/cdh_sg_sentry.html#xd_583c10bfdbd326ba--7f25092b-13fba2465e5--7f93).

**Impala and Sentry**

Authorization processing in Impala is similar to that in Hive. The main difference is caching of privileges. Impala’s Catalog server manages caching schema metadata and propagating it to all Impala server nodes. This Catalog server caches Sentry metadata as well. As a result, authorization validation in Impala happens locally and much faster.

