Sentry Tutorial

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.

Continue reading:

Architecture Overview

Sentry Components

There are components involved in the authorization process:

- **Sentry Server**: The Sentry RPC server manages the authorization metadata. It supports interface 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 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 grant `SELECT` 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 discussed previously, if 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 via MapReduce, Pig applications or metadata access via 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.

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.

Sentry-HDFS Synchronization

Sentry-HDFS authorization is focused on Hive warehouse data - that is, any data that is part of a table in Hive or Impala. The real objective of this integration is to expand the same authorization checks to Hive warehouse data being accessed from any other components such as Pig, MapReduce or Spark. At this point, this feature does not replace HDFS ACLs. Tables that are not associated with Sentry will retain their old ACLs.

The mapping of Sentry privileges to HDFS ACL permissions is as follows:

- **SELECT** privilege -> Read access on the file.
- **INSERT** privilege -> Write access on the file.
- **ALL** privilege -> Read and Write access on the file.

The NameNode loads a Sentry plugin that caches Sentry privileges as well Hive metadata. This helps HDFS to keep file permissions and Hive tables privileges in sync. The Sentry plugin periodically polls the Sentry and Metastore to keep the metadata changes in sync.

For example, if Bob runs a Pig job that is reading from the Sales table data files, Pig will try to get the file handle from HDFS. At that point the Sentry plugin on the NameNode will figure out that the file is part of Hive data and overlay Sentry privileges on top of the file ACLs. As a result, HDFS will enforce the same privileges for this Pig client that Hive would apply for a SQL query.

For HDFS-Sentry synchronization to work, you *must* use the Sentry service, not policy file authorization.

Search and Sentry

Sentry can apply a range of restrictions to various Search tasks, such accessing data or creating collections. These restrictions are consistently applied, regardless of the way users attempt to complete actions. For example, restricting access to data in a collection restricts that access whether queries come from the command line, from a browser, or through the admin console.

With Search, Sentry stores its privilege policies in a policy file (for example, sentry-provider.ini) which is stored in an HDFS location such as `hdfs://ha-nn-uri/user/solr/sentry/sentry-provider.ini`.

Sentry with Search does not support multiple policy files for multiple databases. However, you must use a separate policy file for each Sentry-enabled service. For example, Hive and Search were using policy file authorization, using a combined Hive and Search policy file would result in an invalid configuration and failed authorization on both services.

- Note: While Hive and Impala are compatible with the database-backed Sentry service, Search still uses Sentry's policy file authorization. Note that it is possible for a single cluster to use both, the Sentry service (for Hive and Impala as described above) and Sentry policy files (for Solr).

Authorization Administration

The Sentry server supports APIs to securely manipulate roles and privileges. Both Hive and Impala support SQL statements to manage privileges natively. Sentry assumes that HiveServer2 and Impala run as superusers, usually called `hive` and `impala`. To initiate top-level permissions for Sentry, an admin must login as a superuser.You can use either Beeline or the Impala shell to execute the following sample statement:

```
GRANT ROLE Analyst TO GROUP finance-managers
```

Disabling Hive CLI

To execute Hive queries, you must use Beeline. Hive CLI is not supported with Sentry and therefore its access to the Hive Metastore must be disabled. This is especially necessary if the Hive metastore has sensistive metadata. To do this, modify the **`hadoop.proxyuser.hive.groups`** in **`core-site.xml`** on the Hive Metastore host. For example, to give the **`hive`** user permission to impersonate only members of the **`hive`** and **`hue`** groups, set the property to:

```
<property>
<name>hadoop.proxyuser.hive.groups</name>
<value>hive,hue</value>
</property>
```

More user groups that require access to the Hive Metastore can be added to the comma-separated list as needed.

Using Hue to Manage Sentry Permissions

Hue supports a Security app to manage Sentry authorization. This allows users to explore and change table permissions.

Old Documents

- Getting Started with Apache Sentry (Deprecated)