Overview of Sentry Column Level Privileges

Anne Yu (anneyu@cloudera.com)

Highlights of Apache Sentry Recent Releases

Release 1.5.1

- Released July 2015
- Column-Level Privileges
- Generic Authorization Model (`support Solr Integration with Sentry`)
- Sentry High Availability (HA)
- More Granular Privileges Support (Create, Drop, Alter, Index and Lock)

Release 1.6.0

- Released September 2015
- Sentry Privilege Import/Export
- Generic Authorization Model (`support sqoop2 integration with Sentry`)
- Sentry Webserver Authentication and Authorization

Column Level Privileges - Motivations

1. No need to create views for the purpose of column level authorization:

   Views allow customers to protect access to sensitive data columns, but typically that involves creating a large number of views and the application needs modification to switch views according to who is executing the query.

2. No overhead to manage views for the purpose of column level authorization:

   User cannot have SELECT access to database where views are created. Workaround is to create a new database for views and grant SELECT access to database.

Column Level Privileges - Joint Efforts Across Multiple Teams

The project is developed jointly between Cloudera and Intel and it is also a wide effort in integrating with Hive, Impala, and Hue:

- Sentry - Lenni Kuff, Sravya Tirukkovalur, Steve Ross, Anne Yu (@cloudrea.com)
- Hue - Romain Rigaux - (@cloudera.com), Impala - Dimitris Tsirogiannis (@cloudera.com)
- Sentry/Hive - Xiaomeng Huang, Haifeng Chen, Colin Ma, Guoquan Shen (@intel.com)

Column Level Privileges - Overview

- User grant privileges on individual columns, rather than complete table.

  GRANT SELECT(column_name) ON TABLE table_name TO ROLE role_name;

- User with role granted can do:

  SELECT column_name FROM TABLE table_name;

  SELECT COUNT(column_name) FROM TABLE table_name;

- User can also find out for which column(s) has privileges using metadata operations:

  SHOW DATABASES; - user can see all the database(s) for which he has database, table or column level privileges.

  USE database-name; SHOW TABLES; - user can see all the tables for which he has table or column level privileges.

  SHOW COLUMNS; - user can only list column(s) for which he has privileges.

Column Level Privileges - Integration with HDFS ACLs

If enable Sentry and HDFS ACLs Syncup feature (doc link), grant user with only column level privileges, user will be restricted from directly reading data files from HDFS URI (database, table and partitions). User needs to be explicitly granted with database or table level privileges.

For example,

```
sudo -u hdfs hdfs dfs -getfacl -R /user/hive/warehouse/demo.db
# file: /user/hive/warehouse/demo.db/pageviews/datestamp=2014-09-23/000000_0
# owner: hive
# group: hive
user::rwx
user:hive:rwx
group::---
group:hive:rwx
mask::rwx
other::--x
```

Column Level Privileges - Integrated with Hive/Impala/Hue

- Hive/Impala both support Sentry's column level authorization model
  - SELECT FROM TABLE
  - SHOW COLUMNS
  - SHOW GRANT will add one new column to show column level privilege
  - And a few more metadata operations
  - Others will have the same behavior as before, require database or table level privileges.
- Hue UI also supports column level authorization model
- Integration Sentry and HDFS ACLs Syncup

Column Level Privileges - Integrated into CDH5.5

- Is integrated in Cloudera Production Ready Hadoop Release CDH5.5.0, which will be released in November 2015.

- Will be supported by CDH5.5 Hive, Impala, and Hue.

Column Level Privileges - Demo

- Demo a typical workflow: grant user column level privilege, user can only operate on permitted columns.
- On a managed cluster with Sentry, Hive and other necessary services installed, demo is going to use beeline doing the below:

[admin]

1. CREATE DATABASE test_db;
2. USE test_db;
3. CREATE TABLE test_tb(s STRING, i INT);
4. CREATE ROLE test_role;
5. GRANT SELECT(s) ON TABLE test_tb TO ROLE test_role;
6. GRANT ROLE test_role TO GROUP test_group;

[test_user]

1. SHOW DATABASES;
2. USE test_db;
3. SHOW TABLES;
4. SELECT s FROM test_tb; (should succeed)
5. SELECT i FROM test_tb; (should fail)