SQL Standard Based Hive Authorization

Status of Hive Authorization before Hive 0.13

The default authorization in Hive is not designed with the intent to protect against malicious users accessing data they should not be accessing. It only helps in preventing users from accidentally doing operations they are not supposed to do. It is also incomplete because it does not have authorization checks for many operations including the grant statement. The authorization checks happen during Hive query compilation. But as the user is allowed to execute dfs commands, user-defined functions and shell commands, it is possible to bypass the client security checks.

Hive also has support for storage based authorization, which is commonly used to add authorization to metastore server API calls (see Storage Based Authorization in the Metastore Server). As of Hive 0.12.0 it can be used on the client side as well. While it can protect the metastore against changes by malicious users, it does not support fine grained access control (column or row level).

The default authorization model in Hive can be used to provide fine grained access control by creating views and granting access to views instead of the underlying tables.

SQL Standards Based Hive Authorization (New in Hive 0.13)

The SQL standards based authorization option (introduced in Hive 0.13) provides a third option for authorization in Hive. This is recommended because it allows Hive to be fully SQL compliant in its authorization model without causing backward compatibility issues for current users. As users migrate to this more secure model, the current default authorization could be deprecated.

For an overview of this authorization option, see SQL Standards Based Authorization in HiveServer2.

This authorization mode can be used in conjunction with storage based authorization on the metastore server. Like the current default authorization in Hive, this will also be enforced at query compilation time. To provide security through this option, the client will have to be secured. This can be done by allowing users access only through Hive Server2, and by restricting the user code and non-SQL commands that can be run. The checks will happen against the user who submits the request, but the query will run as the Hive server user. The directories and files for input data would have read access for this Hive server user. For users who don't have the need to protect against malicious users, this could potentially be supported through the Hive command line as well.

The goal of this work has been to comply with the SQL standard as far as possible, but there are deviations from the standard in the implementation. Some deviations were made to make it easier for existing Hive users to migrate to this authorization model, and some were made considering ease of use (in such cases we also looked at what many widely used databases do).

Under this authorization model, users who have access to the Hive CLI, HDFS commands, Pig command line, 'hadoop jar' command, etc., are considered privileged users. In an organization, it is typically only the teams that work on ETL workloads that need such access. These tools don't access the data through HiveServer2, and as a result their access is not authorized through this model. For Hive CLI, Pig, and MapReduce users access to Hive tables can be controlled using storage based authorization enabled on the metastore server.

Most users such as business analysts tend to use SQL and ODBC/JDBC through HiveServer2 and their access can be controlled using this authorization model.

Restrictions on Hive Commands and Statements

Commands such as dfs, add, delete, compile, and reset are disabled when this authorization is enabled.

The set commands used to change Hive configuration are restricted to a smaller safe set. This is controlled using the hive.security.authorization.sqlstd.confwhitelist configuration parameter. If this set needs to be customized, the HiveServer2 administrator can set a value for this configuration parameter in its hive-site.xml.

Privileges to add or drop functions and macros are restricted to the **admin** role.

To enable users to use functions, the ability to create permanent functions has been added. A user in the **admin** role can run commands to create these functions, which all users can then use.

The Hive transform clause is also disabled when this authorization is enabled.

Privileges

SELECT privilege – gives read access to an object.

INSERT privilege – gives ability to add data to an object (table).

UPDATE privilege – gives ability to run update queries on an object (table).

DELETE privilege – gives ability to delete data in an object (table).

ALL PRIVILEGES – gives all privileges (gets translated into all the above privileges).

Objects

- The privileges apply to table and views. The above privileges are not supported on databases.
- Database ownership is considered for certain actions.
- URI is another object in Hive, as Hive allows the use of URI in SQL syntax. The above privileges are not applicable on URI objects. URI used are expected to point to a file/directory in a file system. Authorization is done based on the permissions the user has on the file/directory.

Object Ownership

For certain actions, the ownership of the object (table/view/database) determines if you are authorized to perform the action.

The user who creates the table, view or database becomes its owner. In the case of tables and views, the owner gets all the privileges with grant option.

A role can also be the owner of a database. The "`alter database`" command can be used to set the owner of a database to a role.

Users and Roles

Privileges can be granted to users as well as roles.
Users can belong to one or more roles.

There are two roles with special meaning – **public** and **admin**.
All users belong to the **public** role. You use this role in your grant statement to grant a privilege to all users.

When a user runs a Hive query or command, the privileges granted to the user and her "***current roles***" are checked. The current roles can be seen using the "`show current roles;`" command. All of the user's roles except for the **admin** role will be in the current roles by default, although you can use the "`set role`" command to set a specific role as the current role. See the command descriptions for details.

Users who do the work of a database administrator are expected to be added to the **admin** role.
They have privileges for running additional commands such as "`create role`" and "`drop role`". They can also access objects that they haven't been given explicit access to. However, a user who belongs to the **admin** role needs to run the "`set role`" command before getting the privileges of the **admin** role, as this role is not in current roles by default.

Names of Users and Roles

Role names are case insensitive. That is, “marketing” and “MarkEting” refer to same role.

User names are *case sensitive*. This is because, unlike role names, user names are not managed within Hive. The user can be any user that the hiveserver2 authentication mode supports.

Quoted Identifiers

User and role names may optionally be surrounded by backtick characters (`) when the configuration parameter `hive.support.quoted.identifiers` is set to `column` (default value). All Unicode characters are permitted in the quoted identifiers, with double backticks (``) representing a backtick character. However when `hive.support.quoted.identifiers` is set to `none`, only alphanumeric and underscore characters are permitted in user names and role names.

For details, see HIVE-6013 and Supporting Quoted Identifiers in Column Names.

As of Hive 0.14, user may be optionally surrounded by backtick characters (`) irrespective of the `hive.support.quoted.identifiers` setting.

Role Management Commands

Create Role

```
CREATE ROLE role_name;
```

Creates a new role. Only the **admin** role has privilege for this.

The role names ALL, DEFAULT and NONE are reserved.

Drop Role

```
DROP ROLE role_name;
```

Drops the given role. Only the **admin** role has privilege for this.

Show Current Roles

```
SHOW CURRENT ROLES;
```

Shows the list of the user's current roles. All actions of the user are authorized by looking at the privileges of the user and all current roles of the user.

The default current roles has all roles for the user except for the **admin** role (even if the user belongs to the **admin** role as well).

Any user can run this command.

Set Role

```
SET ROLE (role_name|ALL|NONE);
```

If a role_name is specified, then that role becomes the only role in current roles.

Setting role_name to ALL refreshes the list of current roles (in case new roles were granted to the user) and sets them to the default list of roles.

Setting role_name to NONE will remove all current roles from the current user. (It's introduced in HIVE-11780 and will be included in the upcoming versions 1.3.0 and 1.2.2.)

If a role the user does not belong to is specified as the role_name, it will result in an error.

Show Roles

```
SHOW ROLES;
```

List all currently existing roles.

Only the **admin** role has privilege for this.

Grant Role

```
GRANT role_name [, role_name] ...
TO principal_specification [, principal_specification] ...
[ WITH ADMIN OPTION ];

principal_specification
  : USER user
  | ROLE role
```

Grant one or more roles to other roles or users.

If “WITH ADMIN OPTION” is specified, then the user gets privileges to grant the role to other users/roles.

If the grant statement ends up creating a cycling relationship between roles, the command will fail with an error.

Revoke Role

```
REVOKE [ADMIN OPTION FOR] role_name [, role_name] ...
FROM principal_specification [, principal_specification] ... ;

principal_specification
  : USER user
  | ROLE role
```

Revokes the membership of the roles from the user/roles in the FROM clause.

As of Hive 0.14.0, revoking just the ADMIN OPTION is possible with the use of REVOKE ADMIN OPTION FOR <role> (HIVE-6252).

Show Role Grant

```
SHOW ROLE GRANT (USER|ROLE) principal_name;
```

where `principal_name` is the name of a user or role.

Lists all roles the given user or role has been granted.

Currently any user can run this command. But this is likely to change in future to allow users to see only their own role grants, and additional privileges would be needed to see role grants of other users.

Example of Show Role Grant

```
0: jdbc:hive2://localhost:10000> GRANT role1 TO USER user1;
No rows affected (0.058 seconds)

0: jdbc:hive2://localhost:10000> SHOW ROLE GRANT USER user1;
+---------+---------------+----------------+----------+
|  role   | grant_option  |   grant_time   | grantor  |
+---------+---------------+----------------+----------+
| public  | false         | 0              |          |
| role1   | false         | 1398284083000  | uadmin   |
+---------+---------------+----------------+----------+
```

Show Principals

```
SHOW PRINCIPALS role_name;
```

Lists all roles and users who belong to this role.

Only the **admin** role has privilege for this.

Example of Show Principals

```
0: jdbc:hive2://localhost:10000> SHOW PRINCIPALS role1;
+-----------------+-----------------+---------------+----------+---------------+----------------+
| principal_name  | principal_type  | grant_option  | grantor  | grantor_type  |   grant_time   |
+-----------------+-----------------+---------------+----------+---------------+----------------+
| role2           | ROLE            | false         | uadmin   | USER          | 1398285926000  |
| role3           | ROLE            | true          | uadmin   | USER          | 1398285946000  |
| user1           | USER            | false         | uadmin   | USER          | 1398285977000  |
+-----------------+-----------------+---------------+----------+---------------+----------------+
```

Managing Object Privileges

Object Privilege Commands

Grant

```
GRANT
    priv_type [, priv_type ] ...
    ON table_or_view_name
    TO principal_specification [, principal_specification] ...
    [WITH GRANT OPTION];
```

Revoke

```
REVOKE [GRANT OPTION FOR]
    priv_type [, priv_type ] ...
    ON table_or_view_name
    FROM principal_specification [, principal_specification] ... ;

principal_specification
  : USER user
  | ROLE role

priv_type
  : INSERT | SELECT | UPDATE | DELETE | ALL
```

If a user is granted a privilege WITH GRANT OPTION on a table or view, then the user can also grant/revoke privileges of other users and roles on those objects. As of Hive 0.14.0, the grant option for a privilege can be removed while still keeping the privilege by using REVOKE GRANT OPTION FOR <privilege> (HIVE-7404).

Note that in case of the REVOKE statement, the DROP-BEHAVIOR option of CASCADE is not currently supported (which is in SQL standard). As a result, the revoke statement will not drop any dependent privileges. For details on CASCADE behavior, you can check the Postgres revoke documentation.

Examples:

```
0: jdbc:hive2://localhost:10000/default> grant select on table secured_table to role my_role;
No rows affected (0.046 seconds)

0: jdbc:hive2://localhost:10000/default> revoke update, select on table secured_table from role my_role;
No rows affected (0.028 seconds)
```

Notice that in Hive, unlike in standard SQL, USER or ROLE must be specified in the principal_specification.

Show Grant

```
SHOW GRANT [principal_specification] ON (ALL | [TABLE] table_or_view_name);

principal_specification
  : USER user
  | ROLE role
```

Currently any user can run this command. But this is likely to change in the future to allow users to see only their own privileges, and additional privileges would be needed to see privileges of other users.

Examples of Managing Object Privileges

Find out the privileges user ashutosh has on table hivejiratable:

```
0: jdbc:hive2://localhost:10000> show grant user ashutosh on table hivejiratable;
+-----------+----------------+------------+---------+-----------------+-----------------+------------
+---------------+----------------+----------+
| database  |     table      | partition  | column  | principal_name  | principal_type  | privilege  |
grant_option  |   grant_time   | grantor  |
+-----------+----------------+------------+---------+-----------------+-----------------+------------
+---------------+----------------+----------+
| default   | hivejiratable  |            |         | ashutosh        | USER            | DELETE     |
false         | 1398303419000  | thejas   |
| default   | hivejiratable  |            |         | ashutosh        | USER            | SELECT     |
false         | 1398303407000  | thejas   |
+-----------+----------------+------------+---------+-----------------+-----------------+------------
+---------------+----------------+----------+
```

Find out the privileges user ashutosh has on all objects:

```
0: jdbc:hive2://localhost:10000> show grant user ashutosh on all;
+-----------+-------------------+------------+---------+-----------------+-----------------+------------
+---------------+----------------+----------+
| database  |       table       | partition  | column  | principal_name  | principal_type  | privilege  |
grant_option  |   grant_time   | grantor  |
+-----------+-------------------+------------+---------+-----------------+-----------------+------------
+---------------+----------------+----------+
| default   | hivecontributors  |            |         | ashutosh        | USER            | DELETE     |
false         | 1398303576000  | thejas   |
| default   | hivecontributors  |            |         | ashutosh        | USER            | INSERT     |
false         | 1398303576000  | thejas   |
| default   | hivecontributors  |            |         | ashutosh        | USER            | SELECT     |
false         | 1398303576000  | thejas   |
| default   | hivejiratable     |            |         | ashutosh        | USER            | DELETE     |
false         | 1398303419000  | thejas   |
| default   | hivejiratable     |            |         | ashutosh        | USER            | SELECT     |
false         | 1398303407000  | thejas   |
+-----------+-------------------+------------+---------+-----------------+-----------------+------------
+---------------+----------------+----------+
```

Find out the privileges all users have on table hivejiratable:

```
0: jdbc:hive2://localhost:10000> show grant on table hivejiratable;
+-----------+---------------+------------+---------+-----------------+-----------------+------------
+---------------+---------------+----------+
| database  |     table     | partition  | column  | principal_name  | principal_type  | privilege  |
grant_option  |   grant_time   | grantor  |
+-----------+---------------+------------+---------+-----------------+-----------------+------------
+---------------+---------------+----------+
| default   | hivejiratable  |            |         | ashutosh        | USER            | DELETE     |
false         | 1398303419000  | thejas   |
| default   | hivejiratable  |            |         | ashutosh        | USER            | SELECT     |
false         | 1398303407000  | thejas   |
| default   | hivejiratable  |            |         | navis           | USER            | INSERT     |
false         | 1398303650000  | thejas   |
| default   | hivejiratable  |            |         | navis           | USER            | SELECT     |
false         | 1398303650000  | thejas   |
| default   | hivejiratable  |            |         | public          | ROLE            | SELECT     |
false         | 1398303481000  | thejas   |
| default   | hivejiratable  |            |         | thejas          | USER            | DELETE     |
true          | 1398303380000  | thejas   |
| default   | hivejiratable  |            |         | thejas          | USER            | INSERT     |
true          | 1398303380000  | thejas   |
| default   | hivejiratable  |            |         | thejas          | USER            | SELECT     |
true          | 1398303380000  | thejas   |
| default   | hivejiratable  |            |         | thejas          | USER            | UPDATE     |
true          | 1398303380000  | thejas   |
+-----------+---------------+------------+---------+-----------------+-----------------+------------
+---------------+---------------+----------+
```

Privileges Required for Hive Operations

Codes

Y: Privilege required.

Y + G: Privilege "WITH GRANT OPTION" required.

Action	Select	Insert	Update	Delete	Ownership	Admin	URI Privilege (RWX Permission + Ownership)
CREATE TABLE					Y (of database)		Y (for create external table – the location)
DROP TABLE					Y		
DESCRIBE TABLE	Y						
SHOW PARTITIONS	Y						
ALTER TABLE LOCATION					Y		Y (for new location)
ALTER PARTITION LOCATION					Y		Y (for new partition location)
ALTER TABLE ADD PARTITION		Y					Y (for partition location)
ALTER TABLE DROP PARTITION				Y			
ALTER TABLE (all of them except the ones above)					Y		
TRUNCATE TABLE					Y		
CREATE VIEW	Y + G						
ALTER VIEW PROPERTIES					Y		
ALTER VIEW RENAME					Y		
DROP VIEW PROPERTIES					Y		
DROP VIEW					Y		
ANALYZE TABLE	Y	Y					
SHOW COLUMNS	Y						
SHOW TABLE STATUS	Y						
SHOW TABLE PROPERTIES	Y						
CREATE TABLE AS SELECT	Y (of input)				Y (of database)		
CREATE INDEX					Y (of table)		
DROP INDEX					Y		

ALTER INDEX REBUILD					Y		
ALTER INDEX PROPERTIES					Y		
SELECT	Y						
INSERT		Y		Y (for OVERWRITE)			
UPDATE			Y				
DELETE				Y			
LOAD		Y (output)		Y (output)			Y (input location)
SHOW CREATE TABLE	Y+G						
CREATE FUNCTION						Y	
DROP FUNCTION						Y	
CREATE MACRO						Y	
DROP MACRO						Y	
MSCK (metastore check)						Y	
ALTER DATABASE						Y	
CREATE DATABASE							Y (if custom location specified)
EXPLAIN	Y						
DROP DATABASE					Y		

Version Information

As of Hive 3.0.0 (HIVE-12408), Ownership is not required for the URI Privilege.

Configuration

For Hive 0.13.x

Set the following in hive-site.xml:

- hive.server2.enable.doAs to false.
- hive.users.in.admin.role to the list of comma-separated users who need to be added to **admin** role. Note that a user who belongs to the **admin** role needs to run the "`set role`" command before getting the privileges of the **admin** role, as this role is not in current roles by default.

Start HiveServer2 with the following additional command-line options:

- -hiveconf hive.security.authorization.manager=org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.SQLStdHiveAuthorizerFactory
- -hiveconf hive.security.authorization.enabled=true
- -hiveconf hive.security.authenticator.manager=org.apache.hadoop.hive.ql.security.SessionStateUserAuthenticator
- -hiveconf hive.metastore.uris=' '

For Hive 0.14 and Newer

Set the following in hive-site.xml:

- **hive.server2.enable.doAs** to false.
- **hive.users.in.admin.role** to the list of comma-separated users who need to be added to **admin** role. Note that a user who belongs to the **admin** role needs to run the "`set role`" command before getting the privileges of the **admin** role, as this role is not in current roles by default.
- Add org.apache.hadoop.hive.ql.security.authorization.MetaStoreAuthzAPIAuthorizerEmbedOnly to **hive.security.metastore.authorization.manager**. (It takes a comma separated list, so you can add it along with StorageBasedAuthorization parameter, if you want to enable that as well). This setting disallows any of the authorization api calls to be invoked in a remote metastore. HiveServer2 can be configured to use embedded metastore, and that will allow it to invoke metastore authorization api. Hive cli and any other remote metastore users would be denied authorization when they try to make authorization api calls. This restricts the authorization api to privileged HiveServer2 process. You should also ensure that the metastore rdbms access is restricted to the metastore server and hiverserver2.
- **hive.security.authorization.manager** to org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.SQLStdConfOnlyAuthorizerFactory. This will ensure that any table or views created by hive-cli have default privileges granted for the owner.

Set the following in hiveserver2-site.xml:

- hive.security.authorization.manager=org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.SQLStdHiveAuthorizerFactory
- hive.security.authorization.enabled=true
- hive.security.authenticator.manager=org.apache.hadoop.hive.ql.security.SessionStateUserAuthenticator
- hive.metastore.uris=' '

Known Issues

Hive 0.13

HIVE-6985 – SQL std auth - privileges grants to public role not being honored

HIVE-6919 – Hive sql std auth select query fails on partitioned tables

HIVE-6921 – Index creation fails with SQL std auth turned on

HIVE-6957 – SQL authorization does not work with HS2 binary mode and Kerberos auth

CVE-2014-0228 - Export/Import statement not authorized.

Hive 0.13.1

The known issues noted above under Hive 0.13.0 have been fixed in 0.13.1 release.

References

For information on the SQL standard for security see:

- ISO 9075 Part 1 Framework sections 4.2.6 (Roles), 4.6.11 (Privileges)
- ISO 9075 Part 2 Foundation sections 4.35 (Basic security model) and 12 (Access control)

Troubleshooting

Problem: My user name is in hive.users.in.admin.role in hive-site.xml, but I still get the error that user is not an **admin**. What could be wrong?

> ***Do This:*** Ensure that you have restarted HiveServer2 after a configuration change and that you have used the HiveServer2 command line options as described in Configuration above.
>
> ***Do This:*** Ensure that you have run a '`set role admin;`' command to get the **admin** role.