Data Access 3

Securing Apache Hive

Date of Publish: 2018-07-12

http://docs.hortonworks.com

Contents

Authorizing Apache Hive Access 3

HDFS ACL permissions model 5

Configure storage-based authorization 7
Authorization configuration parameters 8
Storage-based operation permissions 9

Transactional table access 9

External table access 9

Apache Spark access to Apache Hive 10

Remote data access 10

Secure HiveServer using LDAP 11
Secure HiveServer using LDAP over SSL 12
Secure LLAP in HiveServer 13
Connections to Apache Hive 14

Authorizing Apache Hive Access

As administrator, you can choose whether or not to set up Apache Ranger authorization use another authorization model to limi Apache Hive access to approved users.

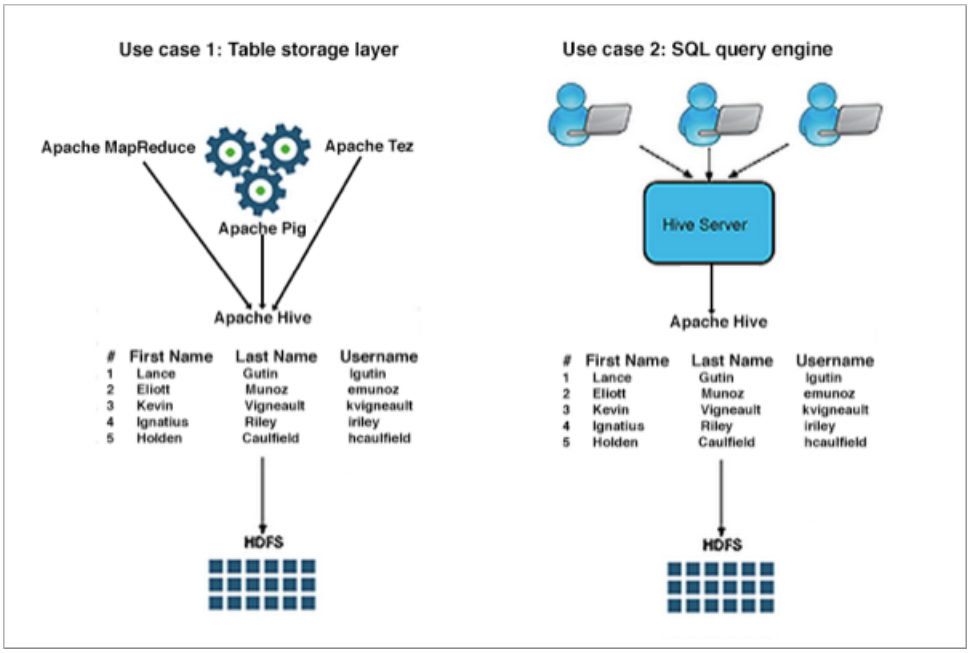
Authorization is the process that checks user permissions to perform select operations, such as creating, reading, and writing data, as well as editing table metadata. Apache Ranger provides centralized authorization for all HDP components and is the recommended for HDP 3.0. It is best to choose an authorization model based on how your organization uses Hive.

There are two primary use cases for Hive:

- Using Hive as a table storage layer

  Many HDP components and underlying technologies, such as Apache Hive, Apache HBase, Apache Pig, Apache MapReduce, and Apache Tez rely on Hive as a table storage layer.
- Using Hive as a SQL query engine

  Hadoop administrators, business analysts, and data scientists use Hive to run SQL queries remotely through a client connecting to Hive through HiveServer. These users often configure a data analysis tool, such as Tableau, to connect to Hive through HiveServer.



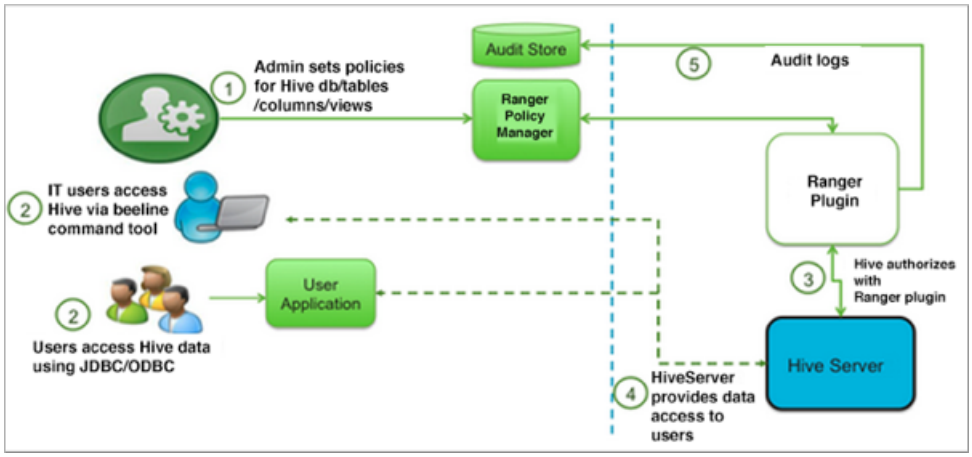
In addition to Apache Ranger, Hive supports storage-based authorization (SBA) for external tables. Ranger and SBA can co-exist in HDP 3.0.0. The following table compares authorization models:

Authorization model	Secure?	Fine-grained authorization (column, row level)	Privilege management using GRANT/REVOKE statements	Centralized management GUI
Apache Ranger	Secure	Yes	Yes	Yes
Storage-based	Secure	No authorization at SQL layer in HiveServer. Provides Metastore server authorization for the Metastore API only.	No. Table privilege based on HDFS permission	No

Authorization model	Secure?	Fine-grained authorization (column, row level)	Privilege management using GRANT/REVOKE statements	Centralized management GUI
Hive default	Not secure. No restriction on which users can run GRANT statements	Yes	Yes	No

Apache Ranger policy authorization

Apache Ranger provides centralized policy management for authorization and auditing of all HDP components, including Hive. All HDP components are installed with a Ranger plugin used to intercept authorization requests for that component, as shown in the following illustration.



Authorizing Hive through Ranger instead of using SBA is highly recommended.

Storage based authorization

As the name implies, storage-based authorization relies on the authorization provided by the storage layer. In the case of an HDP cluster, the storage layer is HDFS, which provides both POSIX and ACL permissions. Hive is one of many HDP components that share storage on HDFS. By enabling this model on the Hive Metastore Server, Hadoop administrators can provide consistent data and metadata authorization. The model controls access to metadata and checks permissions on the corresponding directories of the HDFS file system. Traditional POSIX permissions for the HDFS directories where tables reside determine access to those tables. For example, to alter table metadata for a table stored in HDFS at /warehouse/tablespace/managed/hive, a user must have WRITE permissions on that directory. However, this authorization model doesn't support column-level security.

In addition to the traditional POSIX permissions model, HDFS also provides ACLs, or access control lists, as described in ACLs on HDFS. An ACL consists of a set of ACL entries, and each entry names a specific user or group and grants or denies read, write, and execute permissions for the specified user or group. These ACLs are also based on POSIX specifications, and they are compatible with the traditional POSIX permissions model.

HDFS ACL permissions provide administrators with authentication control over databases, tables, and table partitions on the HDFS file system. For example, an administrator can create a role with a set of grants on specific HDFS tables, then grant the role to a group of users. Roles allow administrators to easily reuse permission grants. Hortonworks recommends relying on POSIX permissions and a small number of ACLs to augment the POSIX permissions for exceptions and edge cases.

A file with an ACL incurs additional memory cost to the NameNode due to the alternate algorithm used for permission checks on such files.

Related Information

Configure a Resource-based Policy: Hive

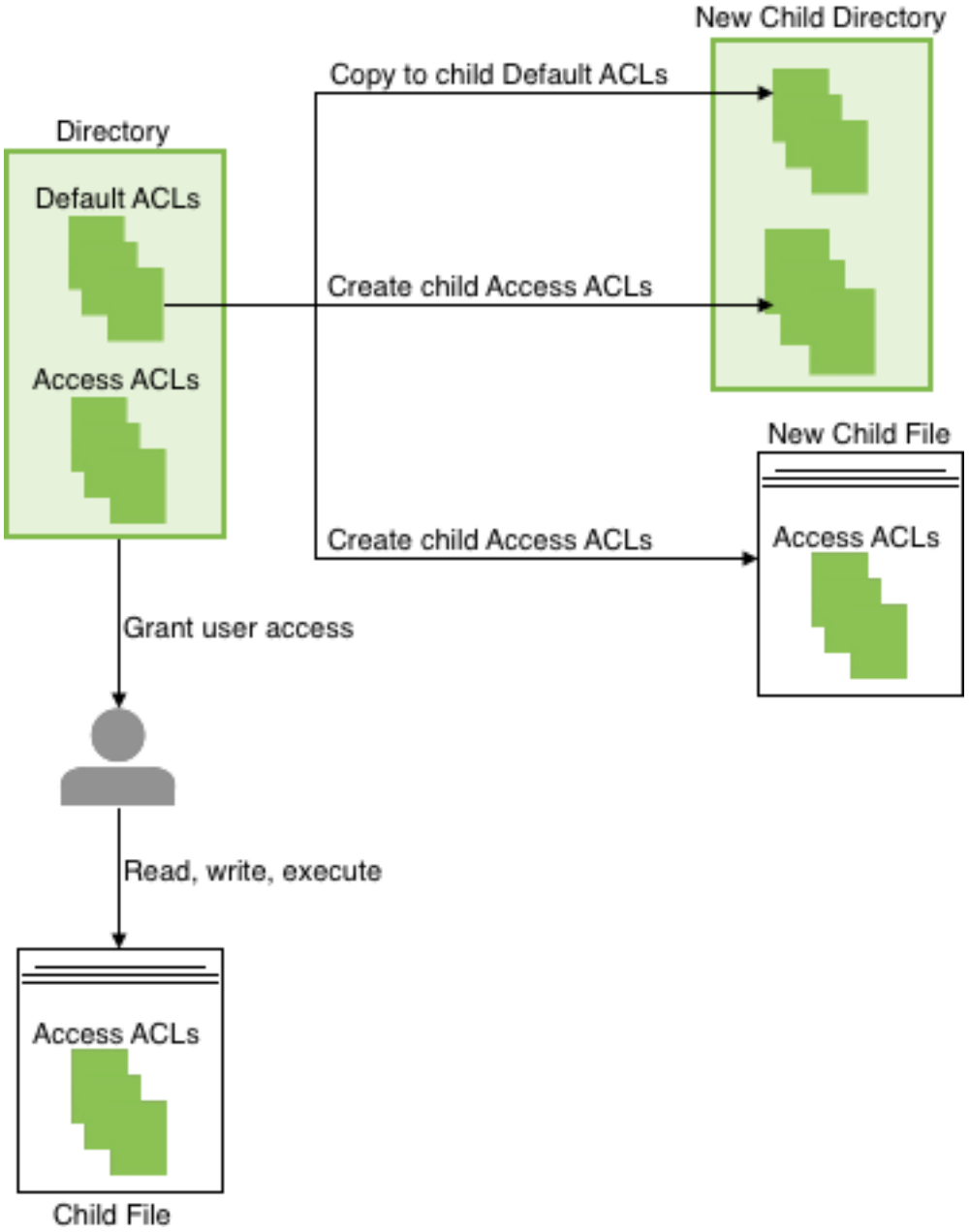
Row-level Filtering and Column Masking in Hive

Start Hive as an end user
Start Hive as the hive user

HDFS ACL permissions model

As administrator, you must understand the permissions model supported in HDP 3.0 and later.

Hive 3 supports the HDFS access control model instead of the past Hive permission inheritance based on the hive.warehouse.subdir.inherit.perms parameter setting. In Hive 3, a directory inherits permissions from the Default ACL. The Default ACL serves as a template from which Access ACLs for subdirectories and files are built. The Access ACL manages user access to child directories and files derived from the Default ACL.



Modifications to the Default ACL of a parent are propagated to the Access ACL or Default ACL of new children only. Existing children are unchanged.

The structures of Default ACLs and Access ACLs are identical:

Entity Type	Entity	Permissions
Owners	Owning user	r-w-x
	Owning group	r-w-x
Named groups and users	marketing	r-w-x
	jane	r-w-x
Unnamed groups and users	other	r-w-x
Not applicable	mask	r-w-x

HDFS permissions

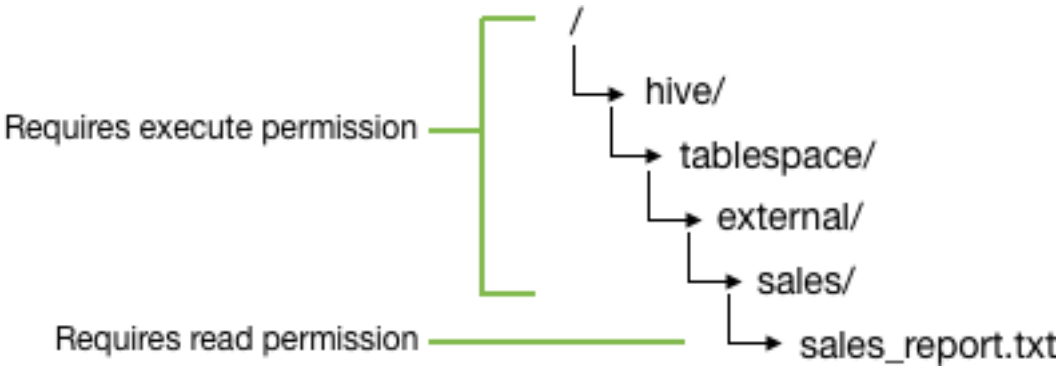
The following table describes read, write, and execute permissions on directories and files:

File		Directory
Read (r)	Able to read a file	Requires r-x to list the directory contents
Write (w)	Able to write or append to a file	Requires r-x to create subdirectories
Execute (e)	Able to parse and run file commands	Requires x to traverse the directory

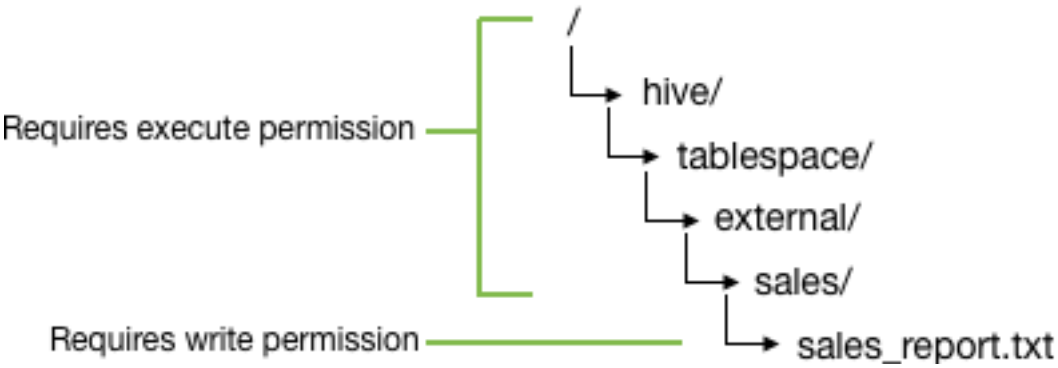
Permission examples

Consider an example in which you want the sales group to access contents of a table for all HiveQL operations. In this case, you must set a Default ACL permissions for the group as - default:group:sales:rwx. The default mask on the directories restricts the permissions granted using Default ACLs. The Default ACL for the hive group is default:group:hive:rwx. This mask gives read, write, and execute access to the hive group and sets permissions on the base directory for databases accessed through Ambari.

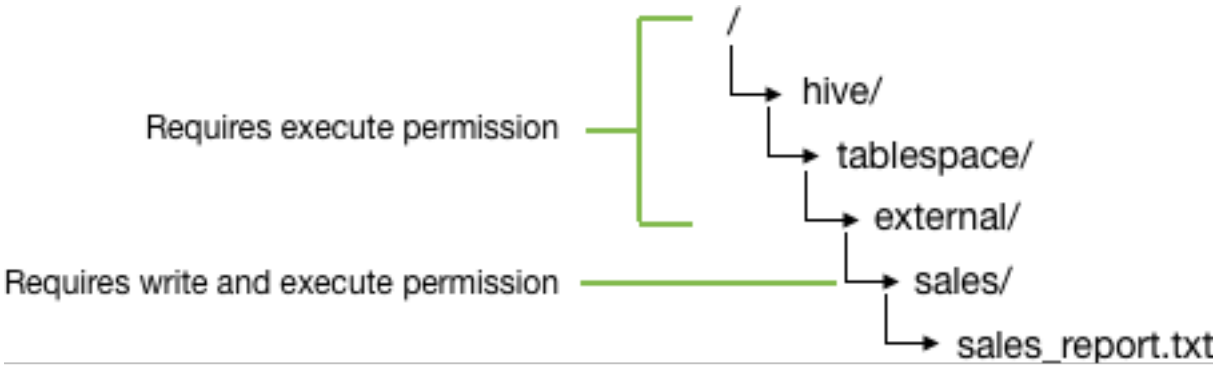
The following example shows directory and file permissions required to read sales_report.txt:



The following example shows directory and file permissions required to append to sales_report.txt:



The following example shows directory permissions required to delete sales_report.txt:



Controlling access to YARN queues using SBA permissions

As administrator, if you do not use the recommended Ranger security, you can configure hive.server2.tez.queue.access=true and the security-based authorization (SBA) doAs impersonation parameter to control access to YARN queues. By setting these parameters, you can allow HiveServer to authorize access to YARN queues for the original user who submitted the query while running the Tez application as the hive user. You configure parameters in one of the following combinations:

- doAs=false (the Hive default and recommended setting)

  Result: The Tez app is submitted as hive, and no access check for the user, for example joe, is performed.
- doAs=true

  When a user, for example joe, submits the query through HiveServer to the YARN queue, for example y_q, the Tez app is started for joe and access to y_q is checked for this user.

hive.server2.tez.queue.access is false by default.

Related Information

Apache Software Foundation HDFS Permissions Guide
HDP 3.0.0 Reference HDFS ACLs

Configure storage-based authorization

You need to set parameters in hive-site.xml to enable storage-based authorization (SBA).

About this task

Hive performs authorization checks on the client, rather than the server when you use SBA. This allows malicious users to circumvent these checks. Some metadata operations do not check for authorization. See Apache JIRA HIVE-3009. DDL statements for managing permissions have no effect on storage-based authorization, but they do not return error messages (HIVE-3010).

Before you begin

- You obtained admin role privileges.

Procedure

1. Set authorization configuration parameters in the hive-site.xml to enable storage-based authorization.

```
<property>
   <name>hive.security.authorization.enabled</name>
   <value>false</value>
</property>
```

```
<property>
   <name>hive.security.authorization.manager</name>

 <value>org.apache.hadoop.hive.ql.security.authorization.StorageBasedAuthorizationPro
value>
</property>

<property>
   <name>hive.server2.enable.doAs</name>
   <value>true</value>
</property>

<property>
   <name>hive.metastore.pre.event.listeners</name>

 <name>org.apache.hadoop.hive.ql.security.authorization.AuthorizationPreEventListener
name>
</property>

<property>
   <name>hive.security.metastore.authorization.manager</name>

 <value>org.apache.hadoop.hive.ql.security.authorization.StorageBasedAuthorizationPro
value>
</property>
```

2. Determine the required permissions of the tables and databases in your environment.
3. Create a table or database in the Hive, then manually modify the POSIX permissions using the HDFS file system commands.

Related Information

Storage-based authorization information on the Apache Wiki

Authorization configuration parameters

Understanding key authorization parameter descriptions help you configure storage-based authorization correctly.

Table 1: Authorization Parameters in hive-site.xml

Configuration Property	Description
hive.security.authorization.enabled	Enables or disables Hive client authorization done as part of query compilation. This property must be set to false in the hive-site.xml file for storage-based authorization, as it is already enabled via checks on metastore API calls.
hive.server2.enable.doAs	Allows Hive queries to be run by the user who submits the query rather than the Hive user. Must be set to true for storage-based access.
hive.metastore.pre.event.listeners	Enables Metastore security. Specify the following value: org.apache.hadoop.hive.ql.security.authorization. AuthorizationPreEventListener.
hive.security.metastore.authorization.manager	The class name of the Hive Metastore authorization manager. Specify the following value for storage-based authorization: org.apache.hadoop.hive.ql.security.authorization. StorageBasedAuthorizationProvider.

Storage-based operation permissions

Table 2: Required Minimum Permissions for Hive Operations

Operation	Database READ Access	Database WRITE Access	Table READ Access	Table WRITE Access
LOAD				X
EXPORT			X	
IMPORT				X
CREATE TABLE		X		
CREATE TABLE AS SELECT		X	X (source table)	
DROP TABLE		X		
SELECT			X	
ALTER TABLE				X
SHOW TABLES	X			

Transactional table access

As administrator, you must set file system permissions or enable the Apache Ranger service for authorization of users who want to work with transactional tables, which are the default and ACID-compliant tables in Hive 3 and later.

ACID tables reside by default in /warehouse/tablespace/managed/hive. Only the Hive service can own and interact with files in this directory. Storage-based authorization (SBA) does not work to give users access ACID tables for the following reasons:

- Limiting users to Hive prevents dirty reads, inconsistencies, and other problems.
- The low-latency analytical processing (LLAP) cache separates data from the storage location, which is incompatible with SBA.

Ranger is the only available authorization mechanism for ACID tables.

External table access

As administrator, you must set up one of several authorization options to allow users to access external tables.

External tables reside by default in /warehouse/tablespace/external on HDFS. To specify some other location of the external table, you need to include the specification in the table creation statement as shown in the following example:

```
CREATE EXTERNAL TABLE my_external_table (a string, b string)
LOCATION '/users/andrena';
```

Hive assigns a default permission of 777 to the hive user, sets a umask to restrict subdirectories, and provides a default ACL to give Hive read and write access to all subdirectories. External tables in HDP 3.0 support the following permissions and authorization models:

- SBA

- SBA and Ranger
- Ranger

You can use the mixed mode, SBA and Ranger, for low-level analytical processing of external tables.

Using the SBA permissions model

You must add Access ACLs to allow groups or users to create databases and tables in the space governed by SBA. You are authorized to query a table if you have file-level access to the underlying data. You configure impersonation in HiveServer to run operations on behalf of an end user. You cannot use LLAP.

Using the SBA and Ranger example

Assume that you are an administrator who creates a sales database and gives the sales group read-write permissions to the sales directory. This includes Default ACLs for the sales group to read from and write to the database. Users in the sales group set doAs=true, and are authorized under SBA to create external tables. Given the ACLs, both Hive and sales users can access all files and partitions.

To restrict certain users from accessing all files and partitions, you can use Ranger. Hive enforces access; however, if you give a sales user fewer options for accessing the tables through SBA, for example by setting a user's HDFS access to tables to read-only, Ranger cannot control that user's access.

Using the Ranger authorization model

If you disable SBA and use only Ranger to give a specific user, who is not in the sales group, permission to create external tables in the sales-report database, the user can log in, use LLAP, and create a database. With Default ACLs in place, sales group users can also access the table.

Related Information

HDFS ACL permissions model

Apache Spark access to Apache Hive

From Apache Spark, you access ACID v2 tables and external tables in Apache Hive 3 using the Hive Warehouse Connector.

The HiveWarehouseConnector library is a Spark library built on top of Apache Arrow for accessing Hive ACID and external tables for reading and writing from Spark.

The Hive Warehouse Connector is optimized for fast transmission of data from low-latency analytical processing (LLAP) to Spark and designed to leverage the LLAP cache. The connector orchestrates a distributed read from LLAP daemons. The read from cache occurs after applying security rules and ACID transformations.

You need low-latency analytical processing (LLAP) to read ACID, or other Hive-managed tables, from Spark. You do not need LLAP to write to ACID, or other managed tables, from Spark. To write to ACID tables with this library, you do not need to deploy LLAP. The HWC library internally uses the Hive Streaming API and LOAD DATA Hive commands to write the data. You do not need LLAP to access external tables from Spark.

Related Information

Hive Warehouse Connector for accessing Apache Spark data

Remote data access

Under certain circumstances can query remote clusters that use a different version of Hive than the version installed on your cluster.

You can query the data on the remote cluster, including the ability to perform WRITE operations from the local cluster.

Examples of Supported Queries

```
CREATE TABLE orders_ctas AS SELECT * FROM orders_ext;
```

```
INSERT INTO orders_ctas SELECT * FROM orders_ext;
```

```
INSERT OVERWRITE TABLE orders_ctas SELECT * FROM orders_ext;
```

Secure HiveServer using LDAP

You can secure the remote client connection to Hive by configuring HiveServer to use authentication with LDAP.

Procedure

1. Add the following properties to the hive-site.xml file to set the server authentication mode to LDAP.

```
<property>
  <name>hive.server2.authentication</name>
    <value>LDAP</value>
</property>

<property>
 <name>hive.server2.authentication.ldap.url</name>
   <value>LDAP_URL</value>
</property>
```

   LDAP_URL is the access URL for your LDAP server. For example, ldap://ldap_host_name@xyz.com:389
2. Add additional properties to the hive-site.xml file, depending on your LDAP service type.
   - Active Directory (AD)
   - Other LDAP service types, such as OpenLDAP

   AD:

```
<property>
 <name>hive.server2.authentication.ldap.Domain</name>
    <value>AD_Domain</value>
</property>
```

   Where AD_Domain is the domain name of the AD server. For example, corp.domain.com.

   Other LDAP service types:

```
<property>
<name>hive.server2.authentication.ldap.baseDN</name>
   <value>LDAP_BaseDN</value>
</property>
```

   Where LDAP_BaseDN is the base LDAP distinguished name for your LDAP server. For example, ou=dev, dc=xyz, dc=com.
3. Test the LDAP authentication by using the Beeline client.
   - If the HiveServer transport mode is binary (hive.server2.transport.mode=binary), use the following syntax:

```
beeline>!connect
```

```
jdbc:hive2://node1:<port>/default
```

- If the HiveServer2 transport mode is HTTP (hive.server2.transport.mode=http) and the Thrift path is cliservice (hive.server2.thrift.http.path=cliservice), use the following syntax:

```
beeline>!connect
jdbc:hive2://node1:<port>/default;transportMode=http;httpPath=cliservice
```

Secure HiveServer using LDAP over SSL

You can secure the remote client connection to Hive by configuring HiveServer to use authentication with LDAP over SSL (LDAPS).

About this task

Two types of certificates can be used for LDAP over SSL with HiveServer2:

- CA Certificates, which are digital certificates that are signed by a Certificate Authority (CA)
- Self-signed certificates

Procedure

1. Add the LDAP authentication property and URL property to the hive-site.xml file to set the server authentication mode to LDAP:

```
<property>
   <name>hive.server2.authentication</name>
      <value>LDAP</value>
</property>

<property>
   <name>hive.server2.authentication.ldap.url</name>
     <value>LDAP_URL</value>
</property>
```

   The LDAP_URL is the access URL for your LDAP server. For example, ldap://ldap_host_name@xyz.com:389.
2. Add additional properties to the hive-site.xml file:
   - If you use Active Directory (AD):

```
<property>
  <name>hive.server2.authentication.ldap.Domain</name>
    <value>AD_Domain</value>
</property>
```

     Where AD_Domain is the domain name of the AD server. For example, corp.domain.com.
   - If you use other LDAP service types including OpenLDAP:

```
<property>
   <name>hive.server2.authentication.ldap.baseDN</name>
     <value>LDAP_BaseDN</value>
</property>
```

     Where LDAP_BaseDN is the base LDAP distinguished name for your LDAP server. For example, ou=dev, dc=xyz, dc=com.
3. Depending on which type of certificate you are using, perform one of the following actions:
   - CA certificate: If you are using a certificate that is signed by a CA, the certificate is already included in the default Java trustStore located at ${JAVA_HOME}/jre/lib/security/cacerts on all of your nodes. If the CA

certificate is not present, you must import the certificate to your Java cacert trustStore using the following command:

```
keytool -import -trustcacerts -alias <MyHiveLdaps> -storepass <password>
 -noprompt -file <myCert>.pem -keystore ${JAVA_HOME}/jre/lib/security/
cacerts
```

If you want to import the CA certificate into another trustStore location, replace ${JAVA_HOME}/jre/lib/security/cacerts with the cacert location that you want to use.

- Self-signed certificate: If you are using a self-signed digital certificate, you must import it into your Java cacert trustStore. For example, if you want to import the certificate to a Java cacert location of /etc/pki/java/cacerts, use the following command to import your self-signed certificate:

```
keytool -import -trustcacerts -alias <MyHiveLdaps> -storepass <password>
 -noprompt -file <myCert>.pem -keystore /etc/pki/java/cacerts
```

4. If your trustStore is not ${JAVA_HOME}/jre/lib/security/cacerts, you must set the HADOOP_OPTS environment variable to point to your CA certificate so that the certificate loads when the HDP platform loads. There is no need to modify the hadoop-env template if you use the default Java trustStore of ${JAVA_HOME}/jre/lib/security/cacerts.
   a) In Ambari, select **Services** > **HDFS** > **Configs** > **Advanced**
   b) Scroll down, and expand the Advanced hadoop-env section.
   c) Add the configuration information to the hadoop-env template text box.

```
export HADOOP_OPTS="-Djava_net_preferIPv4Stack=true
                    -Djavax.net.ssl.trustStore=/etc/pki/java/cacerts
                    -Djavax.net.ssl.trustStorePassword=changeit
 ${HADOOP_OPTS}"
```

   d) Click Save.
5. Restart the HDFS and Hive services.
6. Test the LDAPS authentication.

```
beeline>!connect jdbc:hive2://node1:10000/default
```

Components such as Apache Knox and Apache Ranger do not use the hadoop-env.sh template. The configuration files for these components must be set for LDAPS and manually restarted.

The Beeline client prompts for the user ID and password.

Secure LLAP in HiveServer

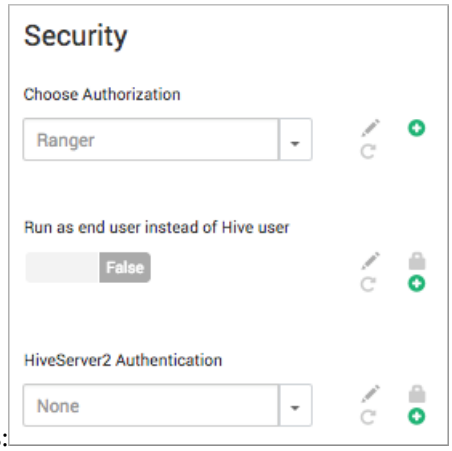
About this task

Hive LLAP shares and caches data across many users like other MPP or database technologies do. Older file-based security controls do not work with Hive and impersonation (doAs=true) is not supported by Hive LLAP. You need to use Apache Ranger, disable impersonation (doAs=false) to secure Hive LLAP, and restrict underlying file access using Ranger policies, so that Hive can access data but unprivileged users cannot.

Procedure

1. Enable Apache Ranger security policies.
2. Set doAs=false in Ambari by setting the Run as end user instead of Hive user to False:

- In Ambari, select **Services** > **Hive** > **Configs**, and set options as follows:



- On the command line, set hive.server2.enable.doAs=false.

Connections to Apache Hive

You can use Beeline, a JDBC, or an ODBC connection to HiveServer.

HiveServer modes of operation

HDP 3.0 supports a number of modes for interacting with Hive, including Ranger-based authorization.

Operating Mode	Description
Embedded	The Beeline client and the Hive installation reside on the same host machine. No TCP connectivity is required.
Remote	Use remote mode to support multiple, concurrent clients executing queries against the same remote Hive installation. Remote transport mode supports authentication with LDAP and Kerberos. It also supports encryption with SSL. TCP connectivity is required.
Operating Mode	Description

Transport Modes

As administrator, you can start HiveServer in one of the following transport modes:

Transport Mode	Description
TCP	HiveServer uses TCP transport for sending and receiving Thrift RPC messages.
HTTP	HiveServer uses HTTP transport for sending and receiving Thrift RPC messages.

Authentication in HiveServer

While running in TCP transport mode, HiveServer supports the following authentication schemes:

Table 3: Authentication Schemes with TCP Transport Mode

Authentication Scheme	Description
Kerberos	A network authentication protocol which operates that uses the concept of 'tickets' to allow nodes in a network to securely identify themselves. Administrators must specify hive.server2.authentication=kerberos in the hive-site.xml configuration file to use this authentication scheme.
LDAP	The Lightweight Directory Access Protocol, an application-layer protocol that uses the concept of 'directory services' to share information across a network. Administrators must specify hive.server2.authentication=ldap in the hive-site.xml configuration file to use this type of authentication.
PAM	Pluggable Authentication Modules, or PAM, allow administrators to integrate multiple authentication schemes into a single API. Administrators must specify hive.server2.authentication=pam in the hive-site.xml configuration file to use this authentication scheme.
Custom	Authentication provided by a custom implementation of the org.apache.hive.service.auth.PasswdAuthenticationProvider interface. The implementing class must be available in the classpath for HiveServer and its name provided as the value of the hive.server2.custom.authentication.class property in the hive-site.xml configuration property file.
None	The Beeline client performs no authentication with HiveServer2. Administrators must specify hive.server2.authentication=none in the hive-site.xml configuration file to use this authentication scheme.