Sentry Service Configuration

The sentry service is a RPC server that stores the authorization metadata in the underlying relational database (eg Derby or MySQL). It provides RPC interfaces to retrieve and manipulate the privileges. It supports secure access to service using Kerberos. The downstream projects like Hive and Impala are client to this service.

Note that the service is just for providing the metadata in the db backed storage. It does NOT handle the actual privilege validation.

Sentry service properties

Config property	Scope	Values	Default	Description
sentry.verify.schema.version	Service	true, false	true	Sentry store will verify the schema version in backed DB with expected version in jar. The service won't start if there's a mismatch
sentry.service.server-max-threads	Service	Number of threads	500	Max worker threads to serve client requests
sentry.service.server-min-threads	Service	Number of threads	10	Min worker threads to serve client requests
sentry.service.allow.connect	Service	comma separated list of users	-	List of users that are allowed to connect to the service (eg Hive, Impala)
sentry.store.jdbc.url	Service	JDBC connection URL	-	JDBC connection URL for the backed DB
sentry.store.jdbc.user	Service	user name	Sentry	Userid for connecting to backend db
sentry.store.jdbc.password	Service	password	Sentry	password for backend JDBC user
sentry.service.server.keytab	Service	Keytab file	-	Keytab for service principal
sentry.service.server.rpc-port	Service	Port #	8038	TCP port number for service
sentry.service.server.rpc-address	Service	TCP bind address	0.0.0.0	TCP interface for service to bind to
sentry.store.jdbc.driver	Service	Backend JDBC driver	org.apache.derby.jdbc.EmbeddedDriver (only when dbtype = derby)	JDBC Driver class for the backed DB
sentry.service.admin.group	Service	Comma separates list of groups		List of groups allowed to make policy updates
sentry.store.group.mapping	Service	`org.apache.sentry.provider.common.HadoopGroupMappingService`	org.apache.sentry.provider.common.HadoopGroupMappingService	Group mapping class for Sentry service. org.apache.sentry.provider.file.LocalGroupMappingService can be used for local group mapping.
sentry.store.group.mapping.resource	Service	Policy file for group mapping		Policy file path for local group mapping, when sentry.store.group.mapping is set to LocalGroupMappingService class.
sentry.service.security.mode	Both	kerberos, none	kerberos	Authentication mode for Sentry service. Currently supports Kerberos and trusted mode
sentry.service.server.principal	Both	Kerberos principal		Service Kerberos principal
sentry.service.client.server.rpc-address	Client	TCP address of the server	-	TCP address of the sentry store server
sentry.service.client.server.rpc-port	Client	Port # of the server	-	Port # of the sentry store server
sentry.service.client.server.rpc-connection-timeout	Client	timeout	200000	RPC connection timeout in milisecs
sentry.metastore.service.users	Client	comma separated list of users		List of service users (eg hive, impala) to bypass the Sentry metastore authorization. These services handle the metadata authorization on their side.
Some common client properties same as file based provider (deprecated names in bracket) Note: Some might need an update SENTRY-295				
`sentry.provider (hive.sentry.provider)`	Client	`org.apache.sentry.provider.file.HadoopGroupResourceAuthorizationProvider`	org.apache.sentry.provider.common.HadoopGroupResourceAuthorizationProvider	Group mapping which should be used at client side
`sentry.hive.server (hive.sentry.server)`	Client	like "server1"	-	Hive Server2 Server identifier like "server1"

sentry.hive.provider.backend	Client	org.apache.sentry.provider.db.SimpleDBProviderBackend org.apache.sentry.provider.file.SimpleFileProviderBackend	org.apache.sentry.provider.file.SimpleFileProviderBackend	Privilege provider to be used, we support file based or db based

Sentry policy store service config file: (sentry-site.xml)

```
  <property>
    <name>sentry.service.security.mode</name>
    <value>none</value>
  </property>
  <property>
    <name>sentry.service.admin.group</name>
    <value>admin1</value>
  </property>
  <property>
    <name>sentry.service.allow.connect</name>
    <value>impala,hive</value>
  </property>
  <property>
    <name>sentry.store.jdbc.url</name>
    <value>jdbc:derby:;databaseName=metastore_db;create=true</value>
  </property>

  <property>
    <name>sentry.store.jdbc.driver</name>
    <value>org.apache.derby.jdbc.EmbeddedDriver</value>
  </property>
```

Sentry policy store client config file: (sentry-site.xml)

```
   <property>
    <name>sentry.service.client.server.rpc-port</name>
    <value>3893</value>
  </property>
  <property>
    <name>sentry.service.client.server.rpc-address</name>
    <value>hostname</value>
  </property>
  <property>
    <name>sentry.service.client.server.rpc-connection-timeout</name>
    <value>200000</value>
  </property>
<!-- Properties required for setting the DB provider-->
  <property>
    <name>sentry.hive.provider.backend</name>
    <value>org.apache.sentry.provider.db.SimpleDBProviderBackend</value>
  </property>
  <property>
    <name>sentry.service.security.mode</name>
    <value>none</value>
  </property>
```

Properties required on Hive to talk to Sentry policy store service: (hive-site.xml)

```
<configuration>
  <property>
    <name>hive.security.authorization.task.factory</name>
    <value>org.apache.sentry.binding.hive.SentryHiveAuthorizationTaskFactoryImpl</value>
  </property>
```

Properties required on Metastore to talk to Sentry policy store service: (hive-site.xml)

```
<property>
  <name>hive.metastore.pre.event.listeners</name>
  <value>org.apache.sentry.binding.metastore.MetastoreAuthzBinding</value>
  <description>list of comma seperated listeners for metastore events.</description>
</property>
```

Starting service :

1. Set HADOOP_HOME
2. Create DB schema for Sentry using Sentry schema tool.

   A built in way to deploy the backed DB schema required for Sentry service. Note that Sentry service by default doesn't initialize the schema. For the initial deployment, you need to run the schematool to initialize the schema. Alternately you can set the service side config property *sentry.verify.schema.version* to false.

   For database other than Derby, implicit deployment is not a good choice. Following option initializes the backend db schema

   ```
   bin/sentry --command schema-tool --conffile <sentry-site.xml> --dbType derby --initSchema
   ```

3. *Start Sentry service*

   ```
   bin/sentry --command service --conffile <sentry-site.xml>
   ```