TLS/SSL Communication Between HMS and Databases

By Aihua Xu

Introduction

With an unencrypted connection between HMS and database, someone with access to the network could watch all your traffic and inspect the data being sent or received between client and server. The databases like Derby, PostgreSQL, Oracle and MySQL currently all support TLS/SSL which will encrypt the data transferred between the client and the server and provide the mechanism to authenticate the client and server.

On the cluster where the data security is a concern, TLS/SSL can be enabled on the database (server side) and then Hive metastore (client side) can be configured to communicate with the database securely with TLS/SSL (referred as SSL only in the following sections).

Enable SSL on Databases

SSL can be enabled differently for different databases and even for different versions of the same database. Also various SSL features are supported for different databases as well. e.g., certain encryption algorithms may not supported for one database. Please follow their docs to set it up and use their client tool to connect in SSL mode to make sure it's enabled properly.

HMS Configuration

HIVE-13044 (Enable TLS encryption to HMS backend database) adds a new hive-site.xml propertity `hive.metastore.dbaccess.ssl.properties` which simplifies the SSL configuration on the HMS side. SSL client configuration can be setup by configuring two `hive-site.xml` properties `javax.jdo.option.ConnectionURL` and `hive.metastore.dbaccess.ssl.properties`.

`javax.jdo.option.ConnectionURL` specifies the connection string for HMS to connect to the database. To enable SSL, the client SSL flag(s) or certain protocol need to add to the connection string.

`hive.metastore.dbaccess.ssl.properties` When SSL is enabled in the connection string, some SSL properties such as key store location or key store password, need to pass in as the system properties. This configuration allows the user to pass in the list of the necessary

SSL properties depending on how the database is configured to secure the connection. e.g., if mutual authentication is needed between the client (HMS) and the server (database), `javax.net.ssl.keyStore` needs to be specified to authenticate the client against the server as well as `javax.net.ssl.trustStore` to authenticate the server against the client.

Here shows some basic configurations for different databases.

MySQL

```
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://myhost/metastore_db?useSSL=true&amp;requireSSL=true</va
    lue>
  </property>
  <property>
    <name>hive.metastore.dbaccess.ssl.properties</name>
    <value>javax.net.ssl.keyStore=/tmp/truststore,javax.net.ssl.keyStorePassword
    =pwd,javax.net.ssl.trustStore=/tmp/truststore,javax.net.ssl.trustStorePasswo
    rd=pwd</value>
  </property>
```

PostgreSQL

```
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://myhost/metastore_db?ssl=true</value>
  </property>
  <property>
    <name>hive.metastore.dbaccess.ssl.properties</name>
    <value>javax.net.ssl.trustStore=/tmp/truststore,javax.net.ssl.trustStorePass
    word=pwd</value>
  </property>
```

Oracle

```
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=ser
vername)(PORT=2484))(CONNECT_DATA=(SERVICE_NAME=servicename))))
</value>
  </property>
  <property>
    <name>hive.metastore.dbaccess.ssl.properties</name>
    <value>javax.net.ssl.trustStore=/tmp/truststore,javax.net.ssl.trustStorePass
    word=pwd,javax.net.ssl.trustStoreType=JKS
```

```
</value>
  </property>
```

Derby

```
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:derby://myhost:1527/metastore_db;ssl=basic</value>
  </property>
  <property>
    <name>hive.metastore.dbaccess.ssl.properties</name>
    <value>javax.net.ssl.trustStore=/tmp/truststore,javax.net.ssl.trustStorePass
    word=pwd</value>
  </property>
```

References

MySQL

- https://dev.mysql.com/doc/refman/5.7/en/using-secure-connections.html
- http://dev.mysql.com/doc/refman/5.7/en/creating-ssl-rsa-files-using-mysql.htm

PostgreSQL

- http://www.postgresql.org/docs/9.1/static/ssl-tcp.html
- https://jdbc.postgresql.org/documentation/91/ssl-client.html

Oracle

- http://www.oracle.com/technetwork/topics/wp-oracle-jdbc-thin-ssl-130128.pdf)

Derby

- https://db.apache.org/derby/docs/10.11/security/csecsslserver.html