Quintessence Labs

APPLICATION NOTE

qCrypt Key Management Server

Using qCrypt as a Key Manager for Oracle TDE 18C/XE, 12C

Table of Contents

Overview

This guide describes the procedure to configure Oracle TDE to use QuintessenceLabs qCrypt key manager as its HSM.

Intended audience: Oracle administrators
Versions:

- QuintessenceLabs qCrypt version 1.8 or later
- QuintessenceLabs qClient-P11 PKCS#11 Adapter
- Oracle TDE 18C/XE, 12C

There are two main tasks: performing a qCrypt pre-setup, and configuring Oracle.

Task 1: qCrypt Pre-setup

Step 1: Creating and Downloading a qCrypt KMIP Connection Pack

The QuintessenceLabs PKCS#11 requires SSL credentials to communicate to qCrypt. This comes in the form of a connection pack that can be created via the web management interface of qCrypt. To create a client connection pack, you will have to do the following on the qCrypt web interface:

1. Create a new KMIP client credential
2. Associate the KMIP client credential with a KMIP client entity

Follow the steps below:

1. Login to the web interface as the 'root' user or a user with enough privilege to create a KMIP client and KMIP client credential.

2. Create a new credential for a KMIP client:
   a. Navigate to the credential generation page by clicking PKI Management → Credentials → generate.
   b. Fill in the entries of the 'Generate New Credential' selecting 'Role' to be 'client'. Do not provide a private key password.
   c. Click generate.

3. Create a new KMIP client (using the previously created credential) by:
   a. Navigating to the 'Add Client' page by clicking KMIP Clients → Clients → add a client.
   b. Fill in form ensuring to:
      i. Not provide a password
      ii. Select the newly created credential
      iii. Setting 'Action Policy' to 'Default All Allowed'
   c. Click 'Add Client' button to add.

4. Download the client connection pack for distribution by clicking 'download connection pack' of the corresponding client shown in the KMIP Clients list view. You will be shown a form. Download the credential by clicking 'Download Connection Pack'.

5. Securely move the downloaded client connection pack to the Oracle instance.

Task 2: Install the QuintessenceLabs PKCS#11 Adapter

The QuintessenceLabs PKCS#11 Adapter needs to be installed for Oracle TDE to use qCrypt as its virtual HSM.

1. Extract the QuintessenceLabs qClient-P11 PKCS#11 Adapter.

```
$ mkdir -p /opt/qlabs/qClient-P11
$ tar -xf <package-name>.tar.gz -C /opt/qlabs/qClient-P11
$ chown -R oracle:oinstall /opt/qlabs/qClient-P11
```

2. Copy the connection pack file into the run directory and change its ownership to the Oracle user.

```
$ cp <connection-pack>.tar /opt/qlabs/qClient-P11
$ chown oracle:oinstall /opt/qlabs/qClient-P11/<connection-pack>.tar
```

3. Now, as the Oracle user, navigate into the run directory and run the initialization script `config.py` providing the user and security officer PIN for the HSM.

```
$ cd /opt/qlabs/qClient-P11
$ python config.py --init --connection-pack <connection-pack>.tar
```

4. In the run directory you can also test connectivity qCrypt by running the test utility to test if the imported connection package is still valid.

```
$ cd /opt/qlabs/qClient-P11
$ python config.py --init
```

Task 3: Configure Oracle to use qCrypt as its HSM

Oracle expects a 3rd-party vendor PKCS#11 driver to be located at predefined location and directory structure. Specifically:

```
/opt/oracle/extapi/[32,64]/hsm/{VENDOR}/{VERSION}/libapiname.ext
```

We will be creating a symbolic file link from the qClient PKCS#11 run directory to Oracles directory.

1. Create new directory to house the QuintessenceLabs qClient PKCS#12 adapter file:

```
$ mkdir -p /opt/oracle/extapi/64/hsm/qClient-P11/1.0.0/
```

2. Create a symbolic link of the `libp6pkcs11.so` file from the QuintessenceLabs PKCS#11 Adapter installation directory to the Oracle HSM directory.

```
$ ln -sf /opt/qlabs/qClient-P11/libp6pkcs11.so  \
/opt/oracle/extapi/64/hsm/qClient-P11/1.0.0/libp6pkcs11.so
```

3. Change the permission of the newly created Oracle HSM directory such that it can be accessible by the Oracle user.

```
$ chown -R oracle:oinstall /opt/oracle/extapi
```

4. For the Oracle user set the set an environmental variable in the `.bashrc`:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/qlabs/qClient-P11
```

Oracle 12C Setup to use a new HSM Encryption Key

1. Set `sqlnet.ora` config file to use the HSM:

```
ENCRYPTION_WALLET_LOCATION=(SOURCE=(METHOD=HSM))
```

The location of the `sqlnet.ora` depends on your specific installation. It could be found in a location such as: `/opt/oracle/product/12c/dbhomeXE/network/admin/sqlnet.ora`.

2. Connect to Oracle as the `sysdba` user.

```
$ sqlplus / as sysdba
```

3. Restart Oracle database for Oracle to detect the new module.

```
SQL> shutdown abort;
SQL> startup;
```

4. Check the status of the wallet.

```
SQL>  select * from v$encryption_wallet;
```

The number of entries you see will be proportional to the number of connections to the qCrypt key manager.

5. Create the encryption key:

```
SQL> ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "password";

keystore altered.
```

Oracle 18C/XE Setup to use a new HSM Encryption Key

1. Set `sqlnet.ora` config file to use the HSM:

```
ENCRYPTION_WALLET_LOCATION=(SOURCE=(METHOD=HSM))
```

The location of the `sqlnet.ora` depends on your specific installation. It could be found is location like `/opt/oracle/product/18c/dbhomeXE/network/admin/sqlnet.ora`.

2. Connect to Oracle as the `sysdba` user.

```
$ sqlplus / as sysdba
```

3. Restart Oracle database for Oracle to detect the new module.

```
SQL> shutdown abort;
SQL> startup;
```

4. Check that status of the wallet:

```
SQL>  select * from v$encryption_wallet;
```

You should see the number of entries proportional to the concurrent connections to the qCrypt key manager.

5. Open the wallet by running the `ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN` command. Provide the corresponding token password set during the installation of the PKCS#11 adapter.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";

keystore altered.
```

6. Check the status of the wallet is open by running `select * from v$encryption_wallet`. Notice that an entry contains `OPEN_NO_MASTER_KEY`.

```
SQL> select * from v$encryption_wallet;

WRL_TYPE
--------------------
WRL_PARAMETER
------------------------------------------------------------------------------
STATUS                WALLET_TYPE         WALLET_OR KEYSTORE FULLY_BAC
------------------------------ -------------------- --------- -------- -
  CON_ID
----------
HSM
OPEN_NO_MASTER_KEY    HSM                 SINGLE     NONE      UNDEFINED
              1

WRL_TYPE
--------------------
WRL_PARAMETER
------------------------------------------------------------------------------
STATUS                WALLET_TYPE         WALLET_OR KEYSTORE FULLY_BAC
------------------------------ -------------------- --------- -------- -
  CON_ID
----------
HSM

CLOSED                UNKNOWN             SINGLE    UNITED    UNDEFINED
              2

WRL_TYPE
--------------------
WRL_PARAMETER
------------------------------------------------------------------------------
STATUS                WALLET_TYPE         WALLET_OR KEYSTORE FULLY_BAC
------------------------------ -------------------- --------- -------- -
  CON_ID
----------
HSM

CLOSED                UNKNOWN             SINGLE    UNITED    UNDEFINED
              3
```

7. Create a key by running the `ADMINISTER KEY MANAGEMENT SET KEY` command:

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "password";

keystore altered.
```

8. Open the wallet:

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";
```

9. Check the status of the wallet again and you will notice that an HSM wallet is now `OPEN`:

```
SQL> select * from v$encryption_wallet;

WRL_TYPE
--------------------
WRL_PARAMETER
------------------------------------------------------------------------------
STATUS                WALLET_TYPE         WALLET_OR KEYSTORE FULLY_BAC
------------------------------ -------------------- --------- -------- -
  CON_ID
----------
HSM

OPEN                  HSM                 SINGLE    NONE      UNDEFINED
              1

WRL_TYPE
```

Task 4: Demonstrate Oracle TDE with a Table

To demonstrate Oracle transparent data encryption, we will create a table that will contain employee information. Table columns are the first name, last name and salary of an employee. The salary column is encrypted.

1. Change to the `system` user or a user that has privileges to create a table:

```
SQL> connect system;
Connected.
```

2. Create an employee database table where the salary column is encrypted.

```
SQL> CREATE TABLE employees ( first_name VARCHAR2(128), last_name
VARCHAR2(128), empID NUMBER, salary NUMBER(6) ENCRYPT);
```

> `ORA-28336: cannot encrypt SYS owned objects` If you see this error then change to the Oracle `system` user by running `connect system` then try creating the table again.

3. View the encrypted table:

```
   SQL> SELECT * from employees;
```

> ORA-28365: wallet is not open: If you see this error then then wallet is closed, open the wallet by running `ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<your-token-password> for Oracle 18c/xe"

4. Insert value into encrypted table.

```
SQL> INSERT INTO employees VALUES ('alice', 'jones', 22, 100);

INSERT INTO employees VALUES ('alice', 'jones', 22, 100);

1 row created.
```

5. View the entry by running `select * from employees;`. If the wallet is open the values should be returned in plaintext:

```
SQL> select * from employees;

FIRST_NAME LAST_NAME    EMPID      SALARY
---------- ----------   ---------- ----------
Alice      jones        22         100
```

6. Let's close the wallet then attempt to read from the encrypted table. Notice that reading from an encrypted table when the wallet is closed will return an error `ORA-28365: wallet is not open` (NEED PRIVILEGES).

   Close the wallet (Oracle 18c/xe):

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "password";

keystore altered.
```

   Close the wallet (Oracle 12c):

```
SQL> ALTER SYSTEM SET ENCRYPTION WALLET CLOSE IDENTIFIED BY "password";

keystore altered.
```

7. Reading an encrypted when the wallet is closed will return an `ORA-28365: wallet is not open` error.

```
SQL> select * from employees;
select * from employees
              *
ERROR at line 1:
ORA-28365: wallet is not open
```

8. Let's open the wallet and read from it again. Notice that the data will be decrypted and returned. Notice to open a wallet requires the KEY MANAGEMENT PRIVILEGES that is typically not possessed by the system user. To grant the KEY MANAGEMENT PRIVILEGE to the system user run the following as the sysdba user:

```
SQL> GRANT ADMINISTER KEY MANAGEMENT TO system;
```

Open the wallet (Oracle 18c/xe) where "password" is the HSM login password:

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";
keystore altered.
```

Open the wallet (Oracle 12) where "password" is the HSM login password.

```
SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY "password";
keystore altered.
```

9. Reading an encrypted table when the wallet is opened will return the requested row:

```
SQL> select * from employees;

FIRST_NAME LAST_NAME    EMPID      SALARY
---------- ----------   ---------- ----------
Alice      jones        22         100
```

Task 5: Inspect Oracle TDE Master Key in qCrypt

You can inspect Oracle's TDE master key in qCrypt's web interface. Navigate to qCrypt's managed object page and search for object names created by KMIP client name used by the PKCS#11 Adapter. Oracle's object names will be identified by the x-attribute of an object under [x-P6R-Cryptoki-LABEL].

Oracle's key id list can be obtained by running on Oracle:

```
SQL> SELECT KEY_ID FROM v$ENCRYPTION_KEYS;
```

Optional: Create an auto-login HSM

An auto-login HSM stores the HSM password in an auto-login Oracle Wallet so the HSM is automatically opened on a request without requiring manual entry of the HSM password.

Note that this configuration reduces the security of the system but supports unattended operations and is useful in deployments where automatic re-login of the HSM is required.

1. Update the `sqlnet.ora` file to use the software keystore type (e.g. METHOD=FILE) and provide the directory of the Oracle Wallet (e.g. DIRECTORY=<oracle_wallet_directory>). You can provide your existing Oracle wallet or create new directory for storing the Oracle wallet.

```
ENCRYPTION_WALLET_LOCATION = (SOURCE=(METHOD=FILE)
(METHOD_DATA=(DIRECTORY=<oracle_wallet_directory>)))
```

For example, if the Oracle Wallet is at `/etc/oracle/wallet` then the sqlnet.ora entry will be:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE=(METHOD=FILE)
(METHOD_DATA=(DIRECTORY=/etc/oracle/wallet)))
```

For the remainder of the steps we will be assuming the Oracle Wallet directory is /etc/oracle/wallet. If the Oracle Wallet directory does not exist, then create it and set the ownership to Oracle:

```
$ mkdir -p /etc/oracle/wallet
$ chown oracle:oinstall /etc/oracle/wallet
```

2. Restart Oracle as the sysdba user:

```
$ sqlplus / as sysdba;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP;
```

3. To configure the auto-login HSM we need to create a software keystore (Oracle Wallet) to store the HSM password. Provide your software keystore password.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/oracle/wallet'
IDENTIFIED BY "software_keystore_password";
```

This will create a `ewallet.p12` file (we do not need to move the auto-login file `ewallet.sso` as it will be overridden in subsequent steps).

4. Open the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
"software_keystore_password";
```

5. Add the HSM password to the software keystore using the 'ADD SECRET' directive. Noting the following:
   - The specific use of single quotes and double quotes. Oracle is very specific about it.
   - The secret to be added (<hsm_password>) is the HSM password and the client name is the `HSM_PASSWORD`. The `HSM_PASSWORD` is an Oracle defined client name that is used to represent the HSM password as a secret in the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET '<hsm_password>' FOR CLIENT
'HSM_PASSWORD' IDENTIFIED BY "<software_keystore_password>" WITH
BACKUP USING '<some_backup_identifier>';
```

For example:

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET 'password' FOR CLIENT
'HSM_PASSWORD' IDENTIFIED BY "software_keystore_password" WITH
BACKUP USING 'backup1';
```

6. Close the software keystore:

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
"software_keystore_password";
```

7. Create a software auto-login keystore providing your previous chosen software keystore password:

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'/etc/oracle/wallet' IDENTIFIED BY "software_keystore_password";
```

This will create an auto-login keystore (cwallet.sso) from the existing password-based software keystore (ewallet.p12).

8. Update the `sqlnet.ora` file to only use the HSM with a software wallet.

```
ENCRYPTION_WALLET_LOCATION = (SOURCE=(METHOD=HSM)
(METHOD_DATA=(DIRECTORY=/etc/oracle/wallet)))
```

9. Restart the database (as the sysdba user or with a user of enough privilege):

```
$ sqlplus / as sysdba;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP;
```

10. Notice that next time a TDE operations occurs, the HSM auto-login keystore opens automatically. Check the wallet:

```
SQL> SELECT * FROM v$ENCRYPTION_WALLET;
```

11. Note that `SELECT * FROM V$ENCRYPTION` will automatically open an auto-login HSM as in any database encryption to the encrypted table by the system user.

Troubleshooting

ORA-28407: Hardware Security Module failed with PKCS#11 error

```
CREATE TABLE employees ( first_name VARCHAR2(128), last_name
VARCHAR2(128), empID NUMBER, salary NUMBER(6) ENCRYPT);
*
ERROR at line 1:
ORA-28407: Hardware Security Module failed with PKCS#11 error
CKR_GENERAL_ERROR(5)
```

Increase the connection timeout in the p6pkcs11.conf configuration file.

ORA-28374: Typed master key not found in wallet

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY “password”;
*
ERROR at line 1:
ORA-28374: typed master key not found in wallet
```

There is no master key set. Create a master key by running:

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY “password”;
```

ORA-28365: Wallet is not open

```
SQL> select * from employees;
select * from employees
*
ERROR at line 1:
ORA-28365: wallet is not open
```

Open the wallet (Oracle 18c/xe):

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY “password”;
keystore altered.
```

Open the wallet (Oracle 12c):

```
SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY “password”;
keystore altered.
```

ORA-28336L: Cannot encrypt SYS owned objects

```
CREATE TABLE employees ( first_name VARCHAR2(128), last_name
VARCHAR2(128), empID NUMBER, salary NUMBER(6) ENCRYPT);

*
ERROR at line 1:
ORA-28336: cannot encrypt SYS owned objects
```

Try logging in as the `system` user by running `connect system` and try again.

```
SQL> connect system;
SQL> [Run command again]
```

Wallet does not stay open

If you find that the wallet does not stay open, Oracle might not have detected the new configuration. Try restarting Oracle and then reopening the wallet.

```
SQL> shutdown immediate;
SQL> startup;
```

About QuintessenceLabs

QuintessenceLabs' portfolio of modular products addresses the most difficult security challenges, helping implement robust security strategies to protect data today and in the future. For more information on QuintessenceLabs' data protection solutions, please visit **www.quintessencelabs.com**

Quintessence Labs

AUSTRALIA
Unit 1, Lower Ground
15 Denison St
Deakin, ACT 2600
+61 2 6260 4922

UNITED STATES
175 Bernal Road
Suite 220
San Jose CA 95119
+1 650 870 9920

www.quintessencelabs.com

Document ID: 4543