Secure External Password Store

An Oracle White Paper
November 2008

ORACLE

The Secure External Password Store

INTRODUCTION

This paper provides step-by-step instructions on how to use the Oracle Database *Secure External Password Store* feature, which was first available with Oracle Database 10g Release 2. This feature does not require the Oracle Advanced Security Option. The secure external password store uses an Oracle Wallet to hold one or more user name/password combinations to run batch processes and other tasks that run without user interaction. The wallet is encrypted using the 3DES algorithm. The secure external password store simplifies large-scale deployments that rely on password credentials for connecting to databases. The best way to envision the password store is as a table with three columns: `TNSALIAS`, `USERNAME`, and `PASSWORD`. The `TNSALIAS` is basically the primary key that maps to a single user name/password combination. In most deployment scenarios, this means creating new `TNSALIAS` entry for each stored credential.

In this example, a reporting tool connects to the database as user '`rep_tool`' to create a nightly report. In order to avoid sharing the password of this account with the developers of the reporting tool, the DSA (Database Security Administrator) decides to store the credentials in a secure external password store. Risk is further reduced because clear-text passwords are no longer hard-coded in application source code, and password management policies are easier enforced without changing application code whenever user names or passwords change.

MODIFYING TNSNAMES.ORA

For example purposes we will use the `tnsnames.ora` file located in `$ORACLE_HOME/network/admin`. Look for an entry called `ORCL` in your `tnsnames.ora` file. Simply cut and paste the `ORCL` entry and rename the copy to `REP_TOOL`. `REP_TOOL` will be the alias used to connect to the database with the credentials stored in the wallet. If you can't find the `ORCL` entry in your `tnsnames.ora` file, simply type in the entry for `REP_TOOL` below.

```
ORCL =
  (DESCRIPTION = (ADDRESS =
      (PROTOCOL = TCP)(HOST = <hostname>)(PORT = 1521))
    (CONNECT_DATA = (SERVICE_NAME = orcl))
  )

REP_TOOL =
  (DESCRIPTION = (ADDRESS =
      (PROTOCOL = TCP)(HOST = <hostname>)(PORT = 1521))
```

```
    (CONNECT_DATA = (SERVICE_NAME = orcl))
  )
```

Test the new alias with:

```
$ tnsping REP_TOOL

TNS Ping Utility for Linux: Version 11.1.0.6.0 - Production on
21-MAR-2008 10:01:51
Used parameter files:
/opt/oracle/product/11.1/db_1/network/admin/sqlnet.ora

Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL =
TCP)(HOST = <hostname>)(PORT = 1521)) (CONNECT_DATA =
(SERVICE_NAME = orcl)))
OK (80 msec)
```

SET THE WALLET LOCATION

Before the wallet can be used to pass credential information to the database for Oracle Net connections, the Oracle Net client must know where to look for the wallet. It is specified in the `sqlnet.ora` file as the `WALLET_LOCATION` parameter and should specify the directory location of the wallet created in the next chapter. This example will create the wallet in the `$ORACLE_HOME/network/admin/` directory, so the following entries will be added to the `sqlnet.ora` file:

```
WALLET_LOCATION =
  (SOURCE = (METHOD = FILE)
   (METHOD_DATA =
    (DIRECTORY = /opt/oracle/product/11.1/db_1/network/admin)))

SQLNET.WALLET_OVERRIDE = TRUE
SSL_CLIENT_AUTHENTICATION = FALSE
```

This setting causes all `sqlplus /@<db_connect_string>` statements to use the information in the wallet at the specified location to authenticate to databases.

CREATING A WALLET

Create a wallet by using the following syntax:

```
mkstore –wrl <wallet_location> –create
```

Change to the directory that will hold the wallet; make sure directory and file permissions are set properly (using a "." in the syntax to specify current working directory):

```
$ mkstore –wrl . –create
Enter password:
Enter password again:
```

The wallet is created:

```
-rw------- 1 oracle oinstall 7340 Mar 21 10:15 cwallet.sso
-rw------- 1 oracle oinstall 7312 Mar 21 10:15 ewallet.p12
```

STORING CREDENTIALS

Create and save database connection credentials in the wallet by using the following syntax:

```
mkstore -wrl <wallet_location> -createCredential
<db_connect_string> <username> <password>

$ mkstore –wrl . -createCredential rep_tool rep_tool
rep_tool_password

Enter password:

Create credential oracle.security.client.connect_string1
```

TESTING THE WALLET CREDENTIALS

Below is an example of connecting to the database using "/"

```
$ sqlplus /@rep_tool

SQL*Plus: Release 11.1.0.6.0 – Production on Fri Mar 21
12:34:22 2008

Copyright © 1982, 2008, Oracle.  All rights reserved

Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 –
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

REP_TOOL>
```

In order to replace the `SQL>` prompt with the current user name, edit

`$ORACLE_HOME/sqlplus/admin/glogin.sql` and add

`set sqlprompt '_user> '` to it.

Simply, `sqlplus /@<db_connect_string>` uses the wallet to lookup the user name and password stored for the matching TNSALIAS, and passes those along to the database.

While it might seem at first glance that this opens up a security hole: "Can't just anyone connect using a `/@<db_connect_string>` that has user name/password stored for it?" The answer lies in where the wallets are stored and who has permission to act upon them. In cases where customers might use wallets to run processing (via CRON or other system scheduler), they should make certain that permissions on the wallet and the wallet directory) are given careful consideration for least-privledge. By locking down the wallets and only running batch-type, "cron" driven processing from trusted clients, this feature protects credentials needed for lights-out processing.

COMMAND-LINE PROXY

Another example for the secure external password store is the following scenario: A **routine batch** program running on the back-end server needs nightly access to the HR application schema, but new security policies have restricted direct access to

the HR application schema. How can the program authenticate to the database using credentials other than the application owner but still have the same level of access?

Solution: Create a separate database account for the program that uses command line proxy with the secure external password store:

```
SQL> grant create session to HRPROC identified by
HRPROC_PASSWORD;
```

Alter the user HR to enable access through the new account.

```
SQL> alter user HR grant connect through HRPROC;
```

Configure the wallet and the tnsnames.ora file:

Add this entry to `tnsnames.ora`:

```
HRPROC =
  (DESCRIPTION = (ADDRESS =
      (PROTOCOL = TCP)(HOST = <hostname>)(PORT = 1521))
    (CONNECT_DATA = (SERVICE_NAME = orcl))
  )
```

Add another set of credentials to your wallet:

```
$ mkstore –wrl . –createCredential HRPROC HRPROC
HRPROC_PASSWORD

Enter password:

Create credential oracle.security.client.connect_string2
```

The batch program can now authenticate to the database using the HRPROC account without specifying credentials on the command line.

```
sqlplus /@HRPROC

HRPROC>
```

HRPROC is allowed to proxy through the HR user, hence the batch program could also use:

```
sqlplus [HR]/@HRPROC
HR>
```

USING THE PASSWORD STORE WITH SSL

If an application is already using SSL for encryption, the `sqlnet.ora` parameter: `SQLNET.AUTHENTICATION.AUTHENTICATION_SERVICES`, specifies SSL and an SSL wallet is created. If this application wants to use the wallet to store credentials to authenticate to databases (instead of the SSL certificate), then those credentials must be stored in the SSL wallet. After SSL authentication, if `SQLNET.WALLET_OVERRIDE = TRUE`, then the user names and passwords from the wallet are used to authenticate to databases. If `SQLNET.WALLET_OVERRIDE = FALSE`, then the SSL certificate is used.

Secure External Password Store
November 2008
Author: Richard Wark
Contributing Authors: Peter Wahl and Paul Needham

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com