Top Tip: Wallets for secure connections in database scripts

By Tim Onions TOdC Limited

Following on from last time's top tip where I praised to the skies two Linux utilities that make the move from Windows to *nix so much less painful this time around I want to share with you a "trick" I fell upon whilst converting MS scheduled tasks to Unix cron jobs, but can be used just as effectively on Windows as it can on Linux/Unix.

Long have I been aware of the security implications of running any kind of batch job in an Oracle environment. Any script you write will need to attach to the database and to do so you need a user name, password (and TNS alias if you are going remote). The *best* solution so far, by far, is to use the OPS$ facility where you leave all troubles of use authentication for Oracle and the operating system to sort out between themselves. The account is defined in a special way - IDENTIFIED EXTERNALLY – and with default installation settings in place you can log into the database, when connected to the database server as OS user FRED, so long as there is an Oracle account identified externally with the name OPS$FRED[1] . Clearly, you need to have the OS account secured otherwise the whole thing is pointless but so long as you do adopt sensible security then this is probably as good as it gets considering none of it costs any extra licensing money (or much effort for that case). However, there are situations where you cannot use this OPS$ functionality – the main case being where you are not connecting to a database on the same machine as the script is being run from. Sure, there is a remote equivalent to OPS$ (REMOTE_OS_AUTHENTICATION) but that comes with its own raft of security risks and it is not a recommended approach as a result – each place you see it mentioned in the Oracle documentation it comes with an associated "health warning".

Having once again this problem staring me in the face I felt it was time, once and for all, that I cracked it. I had read about password repositories and so my first thought was to deploy one of these – and there appeared to be an ideal one in the SourceForge.net OPR project (Oracle Password Repository). Having thought I had this covered – and so early to bed for once - it was not without some little pit of dismay that I received a call from the system's administrator to say that theOPR code would not compile on the particular Linux distribution being used on the project.

Then out of despair came the shining Excalibur of an inbuilt Oracle feature, little known (to me at least and many others I am pretty sure) known as the Secure External Password Store[2]. This is a client side secure encrypted repository of usernames and passwords that can be used directly from a SQL*Net connection as it associates the username and password with a TNS alias string. This means, when it is setup, you have an encrypted

[1] You can use the xxx init.ora setting to change the OPS$ prefix to anything you like or even remove it altogether.

[2] See the Oracle Security Guide (http://download-west.oracle.com/docs/cd/B19306_01/network.102/b14266/cnctslsh.htm) for Oracle's documentation on the password store

username and password safely hidden away in a wallet so you can now connect to any database using a command such as:

```
sqlplus /@remotedb
```

just so long as *remotedb* is a TNS alias in you TNSNAMES.ORA file and that the wallet has an associated entry for *remotedb* too.

Setting up a secure external password really is no trouble what-so-ever (which came as a complete surprise to me as I could well imagine multiple Oracle configuration hoops and server reboots to struggle through). It is a 3 step process, all command line driven, usually conducted from the machine and account that you want to run your scripts from:

Step 1: - Tell TNS you are using a wallet and where it is

This is done by creating a special local .sqlnet.ora file containing the lines (you can place it in the standard sqlnet.ora if you wish but I prefer using the local .sqlnet.ora version so as not to mess with the normal settings):

```
SQLNET.WALLET_OVERRIDE=TRUE
WALLET_LOCATION=(SOURCE=(METHOD=FILE)(METHOD_DATA=(DIRECTORY=/home/etc)))
```

You should specify a sensible, secure location, for you own environment for =/*home/etc.*[3]

Step 2: - Create a wallet

```
mkstore -create -wrl /home/etc
```

The directory specified here needs to match the directory you chose in step 1 (there will be no error if you do not but it simply will not allow you to connect if the two do not match). You will be required to choose, and confirm, a password to use with this wallet. This password it not used when the wallet is used to connect to a remote database, it is the security used around maintenance activities on the wallet itself.

Step 3: - Add each of your required TNS alias, username and passwords to the wallet via the command

```
mkstore -wrl /home/etc -createCredential TNSAlias username password
```

Once again the directory specified here needs to match the directory you chose in step 1 (and no errors are given if you get it wrong). You will also need to supply the password you defined in step 2 in order to make the change to the wallet.

That is all there is to it. Provided you got the directory values consistent between steps 1 and 2 and the TNS alias used in step 3 actually exists in your TNSNAMES.ORA file then you can now connect to each remote database that you created entries for in step3 without having to supply (or even know) the username and password. See, I said it was easy didn't I!

So, using this password store and OPS$ you have all your scripting security issues covered. Well not quite – life in this business is never that simple, it just would not be

[3] The wallet can instead be served from a LPAD server if you have one and do not want to use files.

Oracle if it was. The wallet is not designed to be used in conjunction with the OPS$ feature. So if your .sqlnet.ora specifies a wallet you will no longer be able to connect to the local database, as defined in ORACLE_SID, via sqlplus / - you get an ORA-01017: invalid username/password error, as shown in the screen dump below:

```
[orasched@lnxdb01 ~]$ ls -ltra .sqlnet.ora
-rw-r--r--  1 orasched ora 112 Mar  6 09:54 .sqlnet.ora

[orasched@lnxdb01 ~]$ cat .sqlnet.ora
SQLNET.WALLET_OVERRIDE=TRUE
WALLET_LOCATION=(SOURCE=(METHOD=FILE)(METHOD_DATA=(DIRECTORY=/home/etc)))

[oraschedr@lnxdb01 ~]$ sqlplus /
SQL*Plus: Release 10.2.0.3.0 - Production on Tue Mar 6 16:43:42 2007
Copyright (c) 1982, 2006, Oracle.  All Rights Reserved.

ERROR:
ORA-01017: invalid username/password; logon denied
^C
[orasched@lnxdb01 ~]$ rm .sqlnet.ora

[orasched@lnxdb01 ~]$ sqlplus /
SQL*Plus: Release 10.2.0.3.0 - Production on Tue Mar 6 16:43:57 2007
Copyright (c) 1982, 2006, Oracle.  All Rights Reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - 64bit Production
With the Partitioning, Real Application Clusters and Data Mining options
SQL>
```

This is an annoyance but not insurmountable. Firstly, how often do you need to connect to a local database and a remote database in the same script – not very often, but it happens (as in one of my requirements). It is also quite easy to write a couple of simple commands to move the .sqlnet.ora "out of way" when it is not needed. This is my attempt at doing just that:

```
#Make sure we do not have a local .sqlnet. so we can connect to OPS$ account
find . -type f -maxdepth 1 -name '.sqlnet.ora' -exec mv {} .sqlnet.ora.bak \;
sqlplus -l -s / …
#Put the .sqlnet.ora file back so that can use wallet to connect to a remote db
find . -type f -maxdepth 1 -name '.sqlnet.ora.bak' -exec mv {} .sqlnet.ora \;
sqlplus -l -s /@remotedb.world...
```

That just about wraps it up. Needless to say you need to make sure the wallet files that get created for all of this are secure (there are two of them and they are not tied to the account that created them in any way – I was able copy the wallet files, chmod the access rights and use them on a completely different account). The wallet files are not OS specific either – just for laughs I created a wallet on my Windows2003 Oracle10.2 full client install machine, copied them to my instant client install only laptop and them moved the same files to my Linux database server. In all three cases I was able to use the wallet files to connect to the same remote database without a password. I was particularly surprised (not to say pleased) that instant client was able to use the wallets – which is somewhat perplexing as a laptop with just instant client installed on it has no means of actually creating the wallets in the first place! One slight awkwardness with setting up and using wallets on Windows is the format you use for the directory path. It took a few tries to get it right as it does not use the usual backslashes for windows directories –

instead you need forward slashes. So to use a Windows directory of c:\oracle10g you actually specify c:/oracle10g/.

About the Author

Tim Onions is an independent database consultant with over 15 years' experience with Oracle databases. Tim specialises in the application and database design of high performance systems, as well as tuning and optimisation techniques and can be reached at Tim.Onions@TOdC.co.uk