

# Oracle security done right.

## *Secure database access on the (unix and linux) operating system level.*

*By Frits Hoogland, VX Company*

Security is an important part of modern database administration, and is getting even more important with security audits because of SOX, HIPAA and/or ISO27001, among others. Most Oracle database security products (Virtual Private Database, Database Vault, Audit Vault, to name a few) focus on the database.

An aspect of the database which has hardly changed for at least 15 years is the database installation onto the operating system. The database installation documented by Oracle is meant for optimal flexibility, not optimal security. Still, this is the type of installation found at almost any customers' site. This means the 'oracle' account is used by all administrators (password sharing), and makes administrator actions untraceable (any Oracle database logfile can be deleted or modified with the 'oracle' account).

Despite popular belief, security is not about denying access. Effective security is about reducing access to what is needed, and auditing actions which are deemed important. Just denying (almost) anything doesn't make a site secure, just inflexible and hard to administer.

What do we want to achieve with a secure setup?

### 1. Oracle account locked

The software needs to have an owner. The 'oracle' account is perfect for being the owner of all the Oracle software and files. Because this account is the owner of all Oracle files, and has all access, it should be locked and not be used.

### 2. Personal accounts

In order to be able to identify persons, every administrator needs to have a personal account, and should use this account for administration.

Sharing an account (for example the 'oracle' account) means the password must be known and is shared. Once an administrator is logged on, all Oracle logging can be deleted or manipulated. This means it is hard and unreliable to tell who did what. Not only logging can be manipulated, also executables (through linking) can be modified, which, for example, can remove Database Vault.

Another thing found in the field is administrators using personal accounts solely to enter the system and provide access to the or a shared account. This solves the need for personal accounts at the operating system level, but the problems at

the Oracle level are still present: the Oracle logging can be tampered with and it's hard to tell who has done what.

### 3. The ability to explicitly grant access to users for using Oracle software

By default, the database software is accessible for any user on the server. Oracle 10 is an exception, with previous Oracle versions and Oracle 11 every user can access the Oracle executables.

From a security perspective, it's useful to explicitly grant access to the Oracle database executables and files. This can be accomplished using a group. The Oracle software already uses groups, but hasn't got a group exclusively for this function. Groups already used:

- oinstall  
The purpose of the oinstall group is to provide write access to the *inventory*. Write access is only needed for patching, upgrades or installation. With the Oracle documented installation guide, the oinstall group is the primary group of the 'oracle' user, which means all the files also get this as their group.
- dba  
The purpose of the dba group is to give its members the ability to get SYSDBA access to the database.

This means there isn't a group which has just the function to provide access to the Oracle software. This means we need an additional group. A descriptive group name would be 'oracle'.

### 4. Audited Oracle executable access and read-only Oracle file access

In order to make the execution of important executables accountable, execution of these must be audited. SYSDBA access is audited by the database itself, access to executables which need the 'oracle' owner can be arranged using 'sudo'.

The Oracle files must not be modifiable by the personal accounts. This means the chance of deleting or corrupting any Oracle files by mistake (or on purpose) is minimised.

Securing an Oracle installation this way is only sensible if all other layers involved with using the database are secured too:

#### Operating System

At the operating system level, access other than Oracle needs to be secured too. This especially means access to the 'root' account must be secured with similar

caution. If not, files and audit records are accessible or modifiable again, and the problem solved at the Oracle level is present at 'root' level.

## Database

Inside the database, the 'DBA' role holds the same power as using the 'oracle' account; using the 'DBA' role, it's possible to read and write with the privilege of the userid with which the database processes are running (the 'oracle' account).

This can be solved by disabling usage of the DBA role (the 'SYSTEM' database account uses the DBA role), and granting the necessary rights to database accounts for doing administration, or creating a role with the necessary rights, and granting this role to administrator accounts.

## Installing Oracle in a secure way

1. Create necessary groups and Oracle software owner.

First, the groups and the Oracle software owner need to be created. This has to be done as 'root'.

There are a few things worth noticing here:

- The groupid's and the userid are very high. This is done in order to prevent a group and userid collision, so we can use the same id's on multiple systems. It's very convenient to have fixed id's.
- Oracle related groups which are not used in this article are also created: oper and asm.
- The primary group of 'oracle' is 'oracle', not 'oinstall'. After installation, the group belonging to the installed files is going to be the group of the primary group of the installing user.

```
# groupadd -g 54321 oracle
# groupadd -g 54322 oinstall
# groupadd -g 54323 dba
# groupadd -g 54324 oper
# groupadd -g 54325 asm

# useradd -d /home/oracle -m -g oracle -G dba,asm,users,oinstall -s /bin/bash \
-u 54321 -c "Oracle software owner" oracle
```

2. Installing the database software as 'oracle'

After the creation of the groups and the 'oracle' user, we need to become 'oracle' in order to install the database software. Installing the software itself is

very easy (assuming the settings and required packages as documented are satisfied).

The tricky part is to become 'oracle' from the 'root' session and still be able to use X. In fact, X handling is a prominent part in securing a database installation. At this point, the assumption is the 'root' session has X access (this can be checked by starting a graphical program like 'xclock' or 'xterm' and see if screen is being displayed whilst being 'root')

When working on the console:

```
# env | grep XAUT *1
XAUTHORITY=/tmp/.gdmH0XGSU
# chmod 644 $XAUTHORITY *2
# su oracle *3
$ export XAUTHORITY= /tmp/.gdmH0XGSU *4
```

1. List the file containing the X authorisation (MIT-MAGIC-COOKIE-1)
2. Make cookie readable for the 'oracle' account (any account, in fact)
3. Become 'oracle'.
4. Set X authorisation to the MIT-MAGIC-COOKIE-1 of the root session

When working with an ssh session:

```
# xauth -v *1
Using authority file /root/.Xauthority
xauth> quit *2
# cat /root/.Xauthority > /tmp/xauth.$$ *3
# echo $$ *4
4112
# su oracle *5
$ export XAUTHORITY=/tmp/xauth.4112 *6
```

1. List the current X authorisation file. '~/.Xauthority' is the default file.
2. Quit the 'xauth' tool.
3. Make a copy of the authorization file to a place which is readable for other sessions. Use '\$\$' (parent process id) to make the name unique.
4. List the parent process id, so we can use it when setting XAUTHORITY.
5. Become 'oracle'.
6. Set XAUTHORITY to the file with the correct cookie.

At this point, we have a session as the 'oracle' user, with X. Next up: install the database:

```
$ /oracle/sw/db.10201/database/runInstaller
```

Important settings during the installation:

**Oracle Database 10g Installation - Installation Method**

### Select Installation Method

**Basic Installation**  
Perform full Oracle Database 10g installation with standard configuration options requiring minimal input. This option uses file system for storage, and a single password for all database accounts.

Oracle Home Location:

Installation Type:

UNIX DBA Group:

Create Starter Database (additional 720MB)

Global Database Name:

Database Password:  Confirm Password:

This password is used for the SYS, SYSTEM, SYSMAN, and DBSNMP accounts.

**Advanced Installation**  
Allows advanced selections such as different passwords for the SYS, SYSTEM, SYSMAN, and DBSNMP accounts, database character set, product languages, automated backups, custom installation, and alternative storage options such as Automatic Storage Management.

**ORACLE**

Choose 'dba' as the 'UNIX DBA Group'. The 'Oracle Home Location' is important for the sudo rules.



Choose 'oinstall' at the 'Specify inventory directory and credentials' screen, at 'Specify Operating System group name'.

Finish the installation.

### 3. Remove 'world' access

After installation, the rights on the binaries differ between versions:

Oracle 10

After the installation, almost all files and directories are not world readable. The exceptions are most executables in \$ORACLE\_HOME/bin.

Oracle 11 and probably Oracle 9 and earlier

After the installation, almost anything installed is world accessible. This is also the situation described in some Metalink note's.

In order to make the Oracle files only accessible using a group (the 'oracle' group), we need to remove 'world' access (this has to be executed as 'root'):

```
# chmod -R o-rwx $ORACLE_HOME
```

(replace \$ORACLE\_HOME with the oracle home directory.)

#### 4. Configure sudo

Most of Oracle's executables need to be run as the owner of the database, which is 'oracle' in most cases. Running executables using another user than the current user is done with 'sudo'.

Most Oracle executables use the 'ORACLE\_HOME' environment variable to locate dynamic loadable libraries, message files, etc. In order to be able to use Oracle executables sudo needs to maintain this variable. (sudo resets the environment of the sudo caller because of security) The environment variables which are maintained are set with 'env\_keep'. Add 'ORACLE\_HOME' to the 'env\_keep' list:

(Remember that all sudo configuration needs to be done as 'root'.)

```
# visudo
...
Defaults env_keep = "ORACLE_HOME COLORS DISPLAY ..."
```

Next you need to configure the executables that certain users or groups are allowed to run:

```
# visudo
...
# entries to enable the dba group to do dba activities
%dba ALL = (oracle) NOPASSWD: /oracle/db/*/bin/lsnrctl
%dba ALL = (oracle) NOPASSWD: /oracle/db/*/bin/dbca
%dba ALL = (oracle) NOPASSWD: /oracle/db/*/bin/netca
%dba ALL = (oracle) NOPASSWD: /oracle/db/*/bin/emctl
%dba ALL = (oracle) NOPASSWD: /oracle/db/*/bin/orapwd
%dba ALL = (oracle) NOPASSWD: /oracle/db/*/bin/netmgr
```

Explanation of these entries:

- First field: user/group. This field contains a username or a groupname. A '%' in front of this field indicates it's a groupname.
- Second field: host. This field contains the hostname on which this entry is valid. 'ALL' means any host.
- Third field: '='. Needed for the sudo 'User Specification' entries.

- Fourth field: Runas specification. The (optional) runas specification configures the user account that is allowed to run the command.
- Fifth field: Tag specification. The (optional) tag specification alters the behavior of sudo. By default, sudo requires a user to authenticate itself (again) when using sudo. NOPASSWD disables this behavior.
- Sixth field: Command. This is the command to be executed. The full path is needed in order to point to a specific executable. An '\*' is inserted here so that executables of different versions of the Oracle software can be used with the rule.

This list depends on the security policy for the environment in which this is implemented. The list should only contain executables to be run as 'oracle' only for operational tasks.

## 5. Add users to the host

The last step is to add users to the unix/linux system and grant the necessary privileges through the groups:

Group	Privilege
oracle	Read and/or execute rights to Oracle executables and files.
dba	SYSDBA access to the database.
oinstall	Write access to the inventory

(the other groups, 'oper' and 'asm' are not mentioned here)

Normal users who do not need to access are not granted any Oracle specific groups mentioned in the table above, which means they have no access to any Oracle file or executable.

Normal (non-DBA) users who need access to the database should only be granted 'oracle', which allows these users to access the database.

DBA's should be granted 'oracle' for access to the executables, and 'dba' to be able to do DBA activities. The 'dba' group gives SYSDBA access to the database, but also sudo access to executables.

As mentioned at 4 'Configure sudo', the sudo entries should only cover operational DBA tasks. Any big or complicated task should be done using the 'oracle' account.

Whilst this seems contradictory, it is not: an additional sudo entry, which should be activated by the system administrator only for a single personal account after a change procedure, should be used for that. The purpose of using the 'oracle' account is to simplify tasks like recovery, patch, installation or update, and because

the patching, installation and update requires inventory write access (granted via the oinstall group), which only the 'oracle' account is and should be member of.

```
# visudo
...
# entry for gaining oracle account
#account ALL = (oracle) NOPASSWD: /bin/bash
```

(This entry is commented out, if 'oracle' access is necessary, the hash ('#') needs to be removed, and 'account' should be changed to the username. Needless to say it should be active only for the time needed)

Once an account is configured for starting a shell as 'oracle', it needs to be done in the following way:

RHEL4: `sudo -H -u oracle -s`

RHEL5: `sudo -u oracle -i`

Because only a single personal account should be enabled temporarily to access the 'oracle' account, and because it should be activated by a system administrator (not an Oracle administrator), any occurrences of issues during which a personal account has access to 'oracle' are easily traceable.

### **Audits of the database administrators**

After these settings are in effect, audit logs for database administration tasks are in two places:

- Oracle database stop, start and SYSDBA/SYSOPER access.

Any stop, start and all access as SYSDBA and SYSOPER is always audited by the database. The location of the audit records is default `$ORACLE_HOME/rdbms/audit` or the location specified in the database parameter 'AUDIT\_FILE\_DEST'. Because of the personal accounts, it's easy to see which account accessed, stopped or started the Oracle database. With version 11 of the Oracle database, the first default location is `$ORACLE_BASE/admin/<ORACLE_SID>/adump`, if that location is unreachable, the second default location is (still) `$ORACLE_HOME/rdbms/audit`.

- Sudo

Any sudo usage is logged via the native system logger (syslogd on linux). On linux, syslogd writes logging belonging to the 'authpriv' facility to the '/var/log/secure' file. 'sudo' uses this facility for writing messages about usage.

In order to audit any SQL executed from a session which logged on with the user 'SYS' and/or a session with SYSDBA or SYSOPER privileges, the database parameter: 'AUDIT\_SYS\_OPERATIONS' can be set to 'TRUE'. The audit records are written to 'AUDIT\_FILE\_DEST'.