ABSOLUTE TECHNOLOGIES

www.absolute-tech.com

Auditing the DBA in Oracle Applications: A Guide for Compliance and Audit Managers

By Cameron Larner

Absolute Technologies, Inc.

Table of Contents

Introduction

In these days of Sarbanes-Oxley (SOX) regulations, Governance, Risk and Compliance (GRC) managers for Oracle E-Business Suite have the assignment to develop and maintain proper controls for their information systems that align with best practices so their organizations can attain compliance. These individuals may be well suited to design, approve and enforce business process-related controls, but they may not fully understand how to audit and control technical professionals like Database Administrators (DBA) and Information Technology (IT) Staff. DBAs and IT Staff access and manipulate enterprise data using methods that may be confusing or unknown to managers with less technical knowledge.

Therefore, it is essential to introduce and support appropriate and effective system based controls by creating and maintaining a secure audit trail of transactions and operations within corporate databases that can be overseen by non-technical managers. To accomplish this, those managers and auditors tasked with compliance must gain a better understanding of the DBA's role, privileges, and capabilities, and how to effectively audit DBA activity in the database. *They cannot simply rely on DBAs to audit themselves.* This would violate basic segregation of duties principles. [For the purposes of brevity, references to the term "DBA" shall include DBAs and other IT Staff with administrative or privileged access to the database.]

Intended Audience

As the title suggests, this paper will benefit compliance and audit managers, non-technical managers and business analysts who are tasked with securing Oracle E-Business Suite for SOX and Segregation of Duties compliance purposes, but are not familiar with the general details of the Oracle database, nor with the capabilities and privileges of the DBA. Specifically, this paper speaks to those who have or are implementing controls for application end users where the next step is to design and implement a strategy for deploying controls for DBA and IT staff users.

Background

In the context of recent Sarbanes-Oxley legislation (SOX), external auditors have scrutinized DBA access and have required controls and systematic proof of those controls before certifying SOX or GRC compliance. After all, the systematic controls for application end users have little impact on your DBA's ability to tamper with financial records in the database.

Objectives

To facilitate the implementation of such controls, this paper will familiarize the reader with important attributes of the Oracle Database and the available roles and privileges of an Oracle DBA. It reviews and discusses approaches and mechanisms to limit DBA power, segregate DBA duties, and audit DBA activity. With this information, it is possible to mitigate the risk that a DBA could modify or circumvent end user controls without detection, obstruct audit mechanisms, or compromise the audit trail itself.

DBA and Database Basics

If the reader feels that he or she has an adequate understanding of the significant constructs of the Oracle Database and the primary functions and powers of the DBA within, then the reader may wish to skip ahead to the section entitled, "Approaches to Auditing the DBA."

DBA's Primary Functions

The DBA role encompasses many functions, all of which may be performed by one or many individuals in an organization, utilizing one or many different methods of access to the database. Those smaller organizations that only have one or two DBAs will usually face greater scrutiny by auditors from a segregation of duties point of view because these DBAs typically have unrestricted access to the database, and no peers to validate their work.

The DBA role consists of the following functions:

- Database Creation, Startup and Shutdown
- Application Implementation/Upgrade
- Migration of Data and Program Changes to Production
- Maintenance, Backup & Recovery
- Performance Optimization
- Security/Database User, and often Application User, Management
- Trouble Shooting

Database Objects

Depending on the privileges granted, the DBA may have access to all data in the database. But, what exactly does "all data" mean? The Oracle database is comprised of "Objects." Some of these objects may be familiar, like tables, some may not, like roles. Nevertheless, it is important to have a general understanding of them in order to accurately assess the risk of granting access to these objects to the DBA or anyone.

Database Objects fall into four general categories:

- Those which *contain* data (Business data or Metadata, which is data about data)
- Those which *refer* to other objects
- Those which programmatically *manipulate* data
- Those which *grant access* to data

Here is a brief description of some of the key objects in each category.

Data Containing Objects

Tables

Most E-Business Suite users are familiar with the concept of a table, the database construct that holds records of information segmented by columns. Tables essentially hold all the valuable application (financial and business) content of the database, and thus are the most important object of focus when considering compliance initiatives.

Indexes
An Index contains technical data about a table. They facilitate quick access to an individual record or subset of records in a table. They may also be used to enforce a unique value for a single column or set of columns in the table to prevent duplicate records. The loss or modification of indexes can have a serious impact on system performance.

Sequences
When called, a sequence returns a unique ascending or descending integer from the initial or previously returned integer. It is often used to generate a unique identifier for the primary key column of a table.

Object Referencing Objects

Views
A View is a construct that looks like a table to an end user, and has similar column and row properties containing values, but actually contains no data itself. In contrast, it is merely a logical table based on one or more actual tables or other views. It can provide a user who does not have direct access to a table, indirect access to the table via the view. Views are often used to de-normalize the relational data of several related tables into a single, end user friendly, logical table.

Synonyms
A Synonym is an alias or alternative name for another database object. Typically, it is used to alias an object owned by another user such that the current user need not specify the schema owner of the non-owned object whenever referenced.

Schemas
A Schema is a collection of all the database objects owned by a single user. The Schema name is the same as the user name.

Data Manipulating Objects

Procedures
A Procedure is a stored block of PL/SQL which can be executed by name. PL/SQL is Oracle's programming language which can be used to insert, update and/or delete records from tables, and/or create, alter or drop database objects.

Functions
A Function is similar to a procedure, except that a function returns a single value when executed, and thus may be used in SQL select statements to generate a selected value or a where clause operand value.

Packages
A Package is a collection of related procedures and functions encapsulated in a single object, where each procedure or function is executable as an individual object.

Triggers
A Trigger is a stored block of PL/SQL which is associated with a table or a system level event. Oracle automatically executes the trigger when a specified SQL statement is issued against the table, or when a specified system event occurs. System events include database startup and shutdown, user login and logoff, server error, DDL execution, etc…

Data Access Objects

Users
A User is an account through which an authorized individual can directly connect to the database, and to which database privileges can be assigned. A User which owns database objects may be referred to as a Schema. Thus, there are really three general categories of Users: those which connect to the database to browse or make transactions, those which contain the database objects of an application, and those which do both.

It is worth noting that the database user account defined above, is distinct from an application end user account. For example, Oracle E-Business Suite provides for the creation of an *application user* account by which the end user logs onto the application. This application user does not correspond to a database user, and does not provide direct database access, but rather is used by the application to validate access to the application at the application level. Once validated by the application, the application automatically connects to a master database user account, called APPS in this case, giving the application user indirect access to the database via their secure application connection.

Roles
A Role is a set of privileges that can be granted to users and/or other roles.

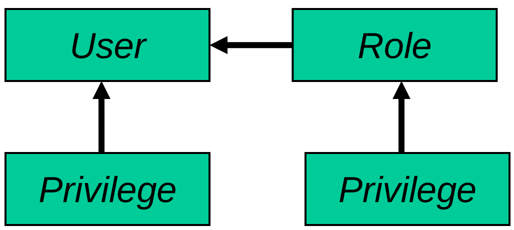
Database Links
A Database Link is an object in the local database, which allows one to connect as a specified user, without the password, to a remote database to access objects in that remote database.

Database Access

The only way to gain access to the database is to connect as a User. However, even if one has a valid User account, the user cannot access individual objects or execute SQL commands without first having been given privileges to do so.

Oracle comes packaged with a wide selection of *privileges*. These privileges are typically granted to the user by a DBA or System Administrator. The ability to grant a privilege to a user is a privilege in and of itself, thus any user with such privilege could administer privileges. The Oracle database comes packaged with several default users, some of which, SYS and SYSTEM for example, have been seeded with such administrative privileges.

Although privileges cannot be created, modified, or deleted as independent entities, they can be bundled as Roles, and granted to or revoked from users as Roles to ease the burden of privilege administration.



Database Operations

When it come to developing controls for access to and use of the database, it’s essential that one understands the basic types of operations (transactions) that may be performed by a DBA or any user.

Here is a review of the general database operations in Oracle:

<u>Select</u>
A user can query and view rows and column values from tables and views.

The selection of data does not change it. You may be concerned about selection if your organization must restrict legally defined sensitive or private information to authorized personnel and exclude DBAs or IT Staff.

<u>DML [Data Manipulation Language]</u>
A user can insert, update, and delete table records.

Using DML, a user can directly modify financial transactions; setups that drive financial transactions; reporting setups; and application user, role and responsibility assignment records. The assignment records govern all security and end user access at the application layer.

<u>DDL [Data Definition Language]</u>

A user can create, alter, and drop database objects. The user can also Grant Privileges to users and roles, and Revoke Privileges.

Using DDL, a user can alter programs stored in the database that are used in the calculation and creation of financial transactions. In fact, a knowledgeable user could introduce "Trojan horse" stored programs which could create fraudulent transactions to the benefit of the user, and which could subsequently remove themselves from the database, leaving no trace of their existence. Additionally, a user could create another temporary user, grant powerful privileges to the user, execute fraudulent or unauthorized transactions as the user, then drop the user.

Startup and Shutdown Database
Database shutdown can be executed in different modes, which if executed inappropriately, could lead to a loss of pending transactions, and interfere with the normal course of business operations.

DBA Access to Oracle / E-Business Suite

There are many ways a DBA can gain access to the Oracle database of which the reader should be aware. Individuals vested with implementing controls for the DBA and IT staff should be keenly aware of each and every entry point and privilege to which these technical individuals may have access. This is the only way to design preventive controls to attain a reasonable assurance of security.

Default Database Users/Schemas
The Oracle Database comes equipped with several default users/schemas, which are initialized with standard, published passwords. It is imperative to security that these passwords are changed immediately after install, and it is highly recommended that they be changed periodically. [Note: For E-Business Suite Releases 11i and 12, both the APPS and APPLSYS users must have identical passwords. These database passwords, along with those from each application module, must be reset at the Application level *and* the database level.]

User Name	Default Password	Description
Oracle Database Administrative Users (Granted DBA Role)		
SYS	**CHANGE_ON_INSTALL**	**Oracle Master Administrator**
SYSTEM	**MANAGER**	**Oracle System Administrator**
CTXSYS	**CTXSYS**	**Oracle InterMedia Text Administrator**
E-Business Suite 11i Database Users		
APPS	**APPS**	**E-Business Suite Master Account**
APPLSYSPUB	**PUB**	**Application User Login Validation**
APPLSYS	**APPS**	**Application Object Library**
GL	**GL**	**General Ledger**
AP	**AP**	**Payables**
AR	**AR**	**Receivables**
PO	**PO**	**Purchasing**
INV	**INV**	**Inventory**
CE	**CE**	**Cash Management**

Partial Table of Default Database Users for Oracle and Oracle E-Business Suite

The most important of these default users is **SYS**, the Oracle Database's master account. It owns the data dictionary, in which all details about all database objects are stored. It is endowed with all privileges, and thus any user who has access to SYS can perform any transaction or operation in the database against any object no matter who owns it.

The second most important of the default users is **SYSTEM**. The SYSTEM user has been granted all database privileges, owns a few objects of its own, can view SYS owned objects, but cannot alter them.

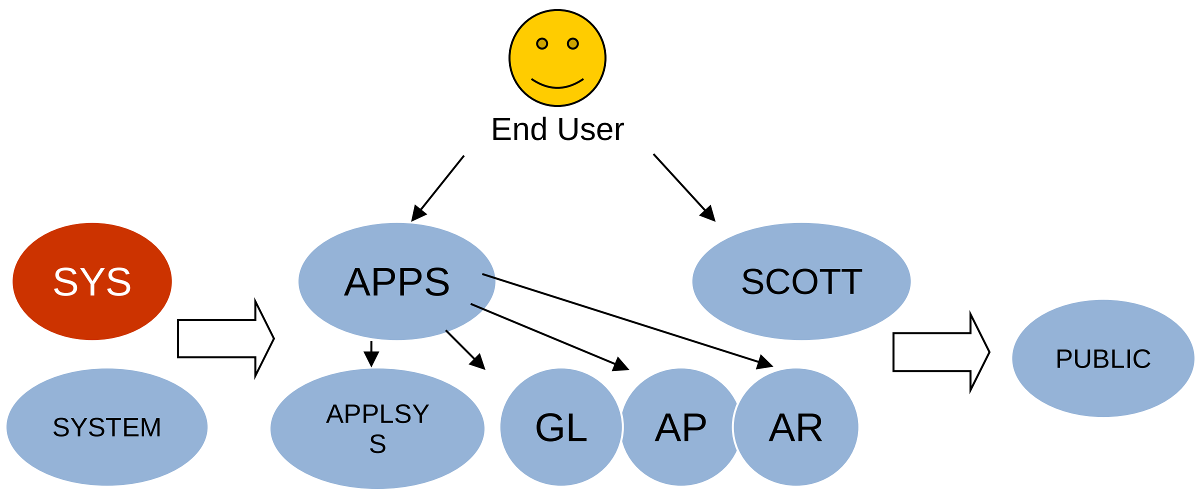
Another important default user is **PUBLIC**. Actually, PUBLIC is really more of a schema than a user. For instance, one cannot connect to the database as "PUBLIC". There is no PUBLIC password. The PUBLIC user acts as a schema repository for particular database objects that are to be granted to all database users. For example, a database user may create a PUBLIC database link or a PUBLIC synonym that is available to all users.

The Oracle E-Business Suite's (EBS) master application database user account is **APPS**. Although the different modules of EBS each install their associated database objects into a module-specific schema, like GL, AP, or AR, the APPS user has been granted privileges to the objects in each of these modules. Thus, using this architecture a single application end user need only connect to the APPS user to gain access to any module in the application. [When an application end user logs on to EBS, the user is connected to the database as APPS.] Once connected to EBS as APPS, EBS uses application level responsibilities, roles, menus, functions and forms to determine what data the application end user can access.

However, any user with the APPS password can connect to APPS directly using a SQL utility like SQL*Plus or Toad, and thus gain unrestricted access to all the database objects within EBS.

Another important default user account for EBS installations is **APPLSYS**. This user/schema owns all the Application Objects Library (AOL) tables and Objects. The AOL module is responsible for, among many other things, the allocation and management of application end user access to the EBS application. Thus, an individual who can connect to the APPLSYS user can create, delete, and modify application users and their responsibility assignments, possibly infringing upon Segregation of Duties rules and controls.

Note that most EBS implementations also have database user accounts in addition to the Oracle or EBS defaults, created for use by third party software or customer initiatives to support custom modules or interfaces. These database accounts should be scrutinized for privileges that grant access to financial or sensitive data in EBS, and may contain such data themselves. Therefore, any security measures mentioned within this paper should also be applied to these types of accounts.



E-Business Suite Partial Schema Map

Administrative Privileges
Although we have discussed a few important default database users, when it comes to the SYS user in an Oracle 9i or greater environment, we must also discuss the newer concept of administrative privileges. Prior to 9i, a user would connect to SYS directly. Now, as an enhanced security measure, a user must specify under which administrative privilege to connect, as SYSDBA or as SYSOPER.

For example, from the SQL prompt:

```
SQL> connect sys as sysdba
```

A user connecting as **SYSDBA** is connected as the SYS user, and essentially has full reign of the database. SYSDBA can create the database, start and shutdown the database, perform archive and recovery tasks, and *access and alter any user's data*.

A user connecting as **SYSOPER**, by contrast, is connected as the PUBLIC user, and has most of the same database management privileges as SYSDBA, but it *cannot view or alter* any other user's data. Thus, the SYSOPER administrative privilege may be an important tool in facilitating required DBA activities while significantly reducing the risk of fraud or controls breach.

Administrative Roles
Distinct from Administrative Privileges are Administrative Roles. These are pre-defined Roles that have been created specifically for administrative/DBA use, without SYS access, which provide further flexibility in achieving security initiatives without tying the hands of the DBA.

Here is a partial list of Admin Roles:

DBA
This Role provides all system privileges with the admin option. The admin option enables the assigned user to grant the same privilege to other users. The SYS and SYSTEM user, by default, have been granted this role.

SELECT_CATALOG_ROLE
This Role provides select access to the Data Dictionary (SYS) views.

EXECUTE_CATALOG_ROLE
This Role provides execute privileges on Data Dictionary (SYS) packages and procedures.

DELETE_CATALOG_ROLE
This Role provides delete access to the SYS owned AUD$ table, which is designed to contain database audit trail records. Oracle comes equipped with a standard audit trail feature which may be activated by setting a database "initialization parameter", and configured using SQL based audit commands to audit certain database transactions. This Role essentially enables the user to purge the audit table. The reader will learn more about this subsequently.

Connection Authentication for Administrators

The functional reader of this paper may find it surprising that an administrator can actually log into the database with admin privileges without a database password. In fact, in most cases you will find that DBAs today rarely use a database password to connect to the database. They rely on operating system (OS) level authentication to connect versus database level authentication via a password file.

As the terms imply, OS authentication enables an administrator with access to the OS of the database server to connect directly to the local database from the server without a password. In contrast, database authentication requires the administrator to provide a valid database password, but the administrator may connect locally or remotely.

Fortunately, OS connection authentication in Oracle is restricted to OS users who belong to one of two account groups, OSDBA ("dba" in Unix) and OSOPER ("oper" in Unix). If an OS user has been assigned to the OSDBA group, that user can connect to Oracle as SYSDBA without providing a database password. Alternatively, an OS user assigned to the OSOPER group can connect to Oracle as SYSOPER without providing a database password.

Furthermore, the Oracle database is installed on the OS server using an OS user account called *oracle*, referred to as the "oracle binary" account. The OS oracle account owns all the files that comprise the Oracle database and it is assigned to the OSDBA group. Therefore, any individual with password access to the oracle account also has direct access to the Oracle database as SYSDBA and does not need the SYSDBA password.

Unfortunately, it is all too common that DBAs are provided with the oracle OS user password. From a security perspective, the problem is twofold: 1) the DBA gains direct access to SYSDBA and 2) there is no way to uniquely identify the individual, since they are only known to the server as "oracle" and to the database as "SYS".

Therefore, it is *very important* to not only restrict and tightly manage database access to SYS, but also to do the same for OS access to the oracle OS account. Whenever possible, create "**named accounts**" at both the OS and database levels, and assign privileges to the named accounts accordingly. Named accounts are created for use by a single individual, often made up of the individual's actual name. For instance, name the user "clarner" or "camlarner" for Cam Larner. As we will later show, this will greatly increase visibility to user connection and transaction activity, as well as discourage fraudulent or unauthorized activity.

Connection authentication to the SYS user can be configured using the database Initialization Parameter called "REMOTE_LOGIN_PASSWORDFILE". This parameter takes one of three arguments:

1. None
   This is the default, which disables the admin database password file, thus forcing OS level authentication, and therefore SYS connections to the database must be via the local server.

2. Exclusive
   With this option, the admin database password file can be used with the local database only. The password file can contain the names of users other than SYS, supporting named account DBA access if the SYS password is not

shared. This option allows SYSDBA and SYSOPER administrative privileges to be granted to individual users so they can connect as themselves from the local server or from a remote server or client. However, from an audit trail perspective, any transactions performed using this connection method will not contain the named account of the user, but rather SYS, which makes identification by user name impossible. [The Oracle Administrator's Guide recommends this option to attain the highest level of security.]

3. Shared
   With this option, the admin database password file may be shared across multiple databases, but the file will only recognize the SYS user entry, so individual named accounts are not supported. This may be an appropriate option if SYS access is restricted to a single DBA who must manage multiple databases. Otherwise, it is not recommended from a security point of view.

Regardless of the option configured, it is very important to note this excerpt from the Oracle Database Administrator's Guide. "Operating system authentication takes precedence over password file authentication. Specifically, if you are a member of the OSDBA or OSOPER group for the operating system, and you connect as SYSDBA or SYSOPER, you will be connected with associated administrative privileges regardless of the *username/password* that you specify."

Another important initialization parameter affecting connection authentication is "O7_DICTIONARY_ACCESSIBILITY". This parameter is designed to provide backward compatibility to version 7 for Oracle database versions 8 and above with respect to SYS and Data Dictionary accessibility. In other words, release 7 of Oracle had lower security standards for Administrative access and setting this parameter to "TRUE" re-enables those lower standards. For example, when set to "TRUE", any database user may be granted full access to SYS objects. If "FALSE", only view access may be granted to other users. Additionally, when set to "TRUE", an administrator can logon to SYS remotely or locally without OS authentication.

Some Oracle E-Business Suite documentation instructs Administrators to set this parameter to "TRUE" during installation and when installing patches. Afterwards, it should be re-set to "FALSE" to ensure maximum security.

If this parameter is set to false and one needs to access objects in the SYS schema, then the user must be granted explicit object privileges. The following roles, which can be granted to the database administrator, also allow access to dictionary objects:

SELECT_CATALOG_ROLE

EXECUTE_CATALOG_ROLE

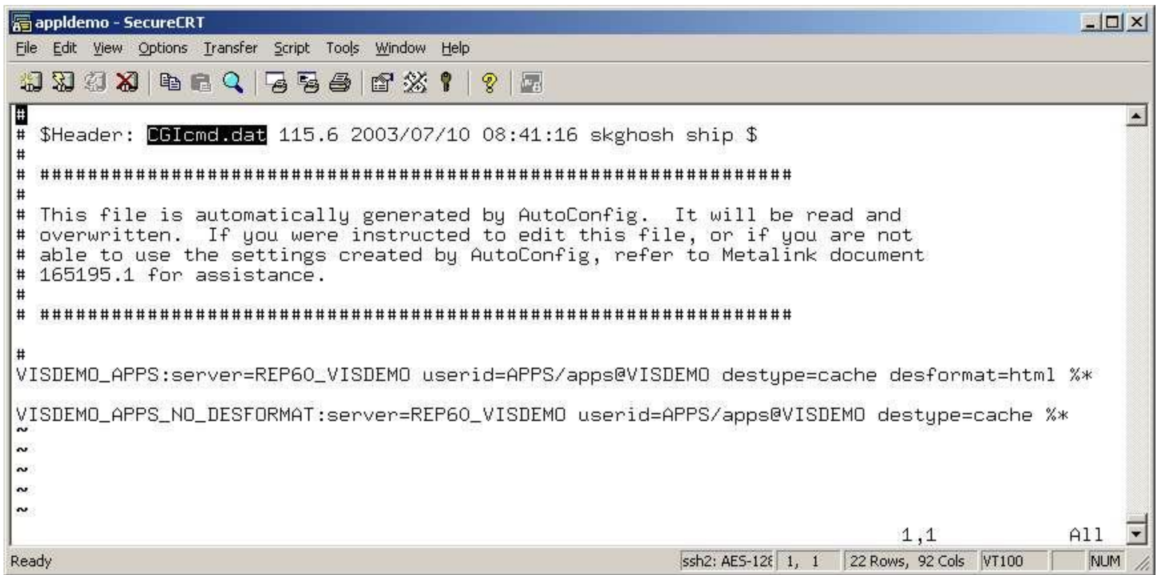
DELETE_CATALOG_ROLE

<u>Application Files with Database Passwords</u>
DBA's and IT staff usually have OS account access to the server upon which the Oracle database is installed. On a UNIX server, all the E-Business Suite source files are typically stored under a single OS user account named "appl" or "appl" concatenated with the instance name, like "appldev", "appltest", or "applprod". In

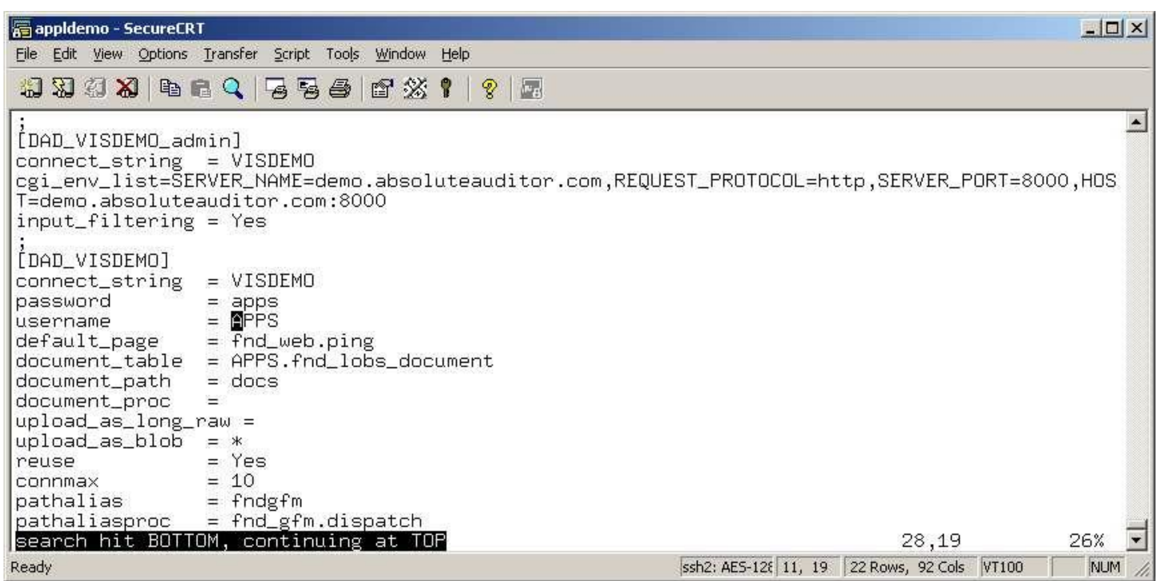
order to manage files required for patches, updates and customizations, DBAs and IT staff will typically have access to the default appl OS user account.

Unfortunately, providing shared access to this default account provides unrestricted access to some directories and files which contain sensitive information or capabilities that should be secured.

For example in EBS R11i, the file $ORACLE_HOME/reports60/server/CGIcmd.dat is used by the application to run reports. It contains a non-encrypted reference to the APPS password, as seen in the example below.



Another R11i example is the file $IAS_ORACLE_HOME/Apache/modplsql/cfg/wdbsvr.app, which is used by the application web server to store database connection descriptors, including various user and password entries including the APPS password.



Fortunately, Oracle has removed these files and thus their reference to the APPS password in R12. However, newer standard EBS functions, like `APPS.DECRYPT_PIN_FUNC`, provide DBAs with the capability to derive the APPS password at will.

Since DBAs are the folks who administer the application, it is understandable that they would have password access to the APPS user. However, is it necessary? Perhaps a *named account* with privileges similar to or the same as APPS would be worthwhile? This would help increase visibility to the individual user and help distinguish those transactions performed by the *application* as APPS verses a *DBA* connected as APPS.

Nevertheless, it may be prudent also to secure the appl OS user account from shared usage. Named accounts can also be created at the OS level, and discriminately given access to necessary files and directories using OS groups.

Additionally, in some cases it may be possible to encrypt and/or obfuscate passwords listed in files to prevent them from falling into unauthorized hands. Discuss these approaches with your Oracle support representative and your System Administrator.

Application Access

A DBA or IT staff member that is privy to the **APPS password** can essentially access, and/or insert, modify or delete any object within the domain of the APPS user, which covers all the application modules within E-Business Suite. The holder of such, can connect directly to the database using a SQL tool like SQL*Plus, could grant himself desired responsibilities, like System Administrator, or create a dummy "unnamed" application user, assign responsibilities to it, and perform transactions that would be very difficult to trace back.

Additionally, if the user also has the SYSTEM password, he could use the $FND_TOP/bin/**FNDCPASS** executable to change application user passwords and log on as someone else. This executable is usually owned by the appl OS user account, and requires the APPS and SYSTEM passwords.

On a more subtle level, if the application "**Examine**" utility is enabled, the user can use the APPS password to login to examine and update fields within forms without detection. The Examine utility is managed via the E-Business Suite System Administrator based profile option called "Utilities:Diagnostics".

Another possible unauthorized access point is **Oracle Alert**, and any other E-Business Suite module with a feature that allows one to enter and execute DML SQL. If a user has been granted the Oracle Alert responsibility, they need not know the APPS password to alter APPS data. When one defines an Oracle Alert, one may enter a SQL statement, which may select, update, insert, or delete records from any table under the APPS domain. When Alert executes the statement, it is already connected as APPS. The user could cover the trail of the unauthorized script afterwards by deleting the Alert log and the Alert itself from within the Oracle Alert module. From a database transaction perspective, any changes to tables will have been made by the generic APPS user, and will have no valid application user level identification such as last_updated_by or last_update_date.

Note that several other E-Business Suite forms allow the execution of user-defined SQL. Metalink Note 189367.1 contains a list of these forms.

Approaches to Auditing the DBA

When it comes to auditing the DBA or any database user, there are many standard options available in the Oracle database. Each approach has advantages and disadvantages. However none offer tamper proof mechanisms, in and of themselves, which can secure the audit trail from DBA access and manipulation. Additional custom and/or third party preventive measures and mechanisms are necessary to mitigate the risk that the DBA may violate audit trail security.

SQL Audit

The Oracle database comes equipped with a standard audit trail feature. To activate the feature, the initialization parameter named "AUDIT_TRAIL" must be set to "TRUE" or "DB" to maintain the audit trail records in the database table owned by SYS called AUD$. Alternatively, it may be set "OS" to maintain the audit trail in files located in the OS directory specified by the initialization parameter called "AUDIT_FILE_DEST". These parameters must be set in the init.ora file, and the database must be shutdown and re-started for them to become effective.

Once activated, SQL audit commands must be issued by a user who has been granted the "audit system" privilege to specify what is to be audited. SYS, SYSTEM and users granted the DBA role have this privilege by default.

[Note: SQL Audit is a database level feature, and not the same mechanism as that called "Audit Trail" provided in the System Administrator responsibility of E-Business Suite. The application level Audit Trail feature relies upon Database Trigger technology, as described in the following section.]

Four basic types of transactions may be audited.

- **Session Connection**
  Audit when a user logs in or out of the database. [Note: Connections using System Administration privileges, SYSDBA and SYSOPER, are always logged to the OS "AUDIT_FILE_DEST" directory even when the audit trail is disabled.]

  Example: SQL> audit session [by user];

- **Statement**
  Audit when a user executes a specified DML or DDL statement against any object.

  Examples: SQL> audit update table, delete table [by user];

- **Privilege**
  Audit when a user executes any statement using a specified privilege.

  Example: SQL> audit delete any table [by user];

- **Object**
  Audit when a user executes any statement on specific object.

Example: SQL> audit select on hr. per_all_people_f;

An audit trail record from the view DBA_AUDIT_TRAIL, based on the SYS. AUD$ table, will contain the following information:

- os_username - Operating system login user name of audited transaction
- username - Database User name of audited transaction
- userhost - Client host machine name
- terminal - Identifier for the user's terminal
- TIMESTAMP - Date/Time of the creation of the audit trail entry (Date/Time of the user's logon for entries created by AUDIT SESSION) in session's time zone
- Owner - Creator of object affected by the action
- obj_name - Name of the schema object accessed
- action - Numeric action type code. The corresponding name of the action type (CREATE TABLE, INSERT, etc.) is in the column ACTION_NAME
- action_name - Name of the action type corresponding to the numeric code in ACTION
- new_owner - The owner of the object named in the NEW_NAME column
- new_name - New name of object after RENAME, or name of underlying object (e.g. CREATE INDEX owner.obj_name ON new_owner.new_name)
- obj_privilege - Object privileges granted/revoked by a GRANT/REVOKE statement
- sys_privilege - System privileges granted/revoked by a GRANT/REVOKE statement
- admin_option - If role/sys_priv was granted WITH ADMIN OPTON,
- grantee - The name of the grantee specified in a GRANT/REVOKE statement
- audit_option - Auditing option set with the audit statement
- ses_actions - Session summary. A string of 12 characters, one for each action type, in this order: Alter, Audit, Comment, Delete, Grant, Index, Insert, Lock, Rename, Select, Update, Flashback. Values: "-" = None, "S" = Success, "F" = Failure, "B" = Both
- logoff_time - Timestamp for user logoff
- logoff_lread - Logical reads for the session
- logoff_pread - Physical reads for the session
- logoff_lwrite - Logical writes for the session
- logoff_dlock - Deadlocks detected during the session
- comment_text - Text comment on the audit trail entry. Also indicates how the user was authenticated
- sessionid - Numeric ID for each Oracle session

- entryid - Numeric ID for each audit trail entry in the session
- statementid - Numeric ID for each statement run (a statement may cause many actions)
- returncode - Oracle error code generated by the action. Zero if the action succeeded
- priv_used - System privilege used to execute the action
- client_id - Client identifier in each Oracle session
- econtext_id - Execution Context Identifier for each action
- session_cpu - Amount of cpu time used by each Oracle session
- extended_timestamp - Timestamp of the creation of audit trail entry (Timestamp of the user's logon for entries created by AUDIT SESSION) in session's time zone
- proxy_sessionid - Proxy session serial number, if enterprise user has logged through proxy mechanism
- global_uid - Global user identifier for the user, if the user had logged in as enterprise user
- instance_number - Instance number as specified in the initialization parameter file ''init.ora''
- os_process - Operating System process identifier of the Oracle server process
- transactionid - Transaction identifier of the transaction in which the object is accessed or modified
- SCN - SCN (System Change Number) of the query
- sql_bind - Bind variable data of the query
- sql_text - SQL text of the query
- obj_edition_name - Edition containing audited object
- dbid - Database Identifier of the audited database

Unfortunately, a few issues with SQL Audit limit its usefulness as a compliance audit tool for auditors.

1. The AUD$ audit table is owned by SYS, and readily accessible to database users with DBA grade privileges. This creates a segregation of duties **red flag**, when the purpose of using such a mechanism is to audit the very DBAs who control it. It is akin to the wolf guarding the hen house.
2. The output of such auditing is often too vague to harvest details that reveal the *impact* the audited transaction had on the database. For instance, when a transaction updates the payroll table, SQL audit does not provide which columns, if any, where changed, not to mention the before and after values of the changed columns. All it provides is the user who performed the PAYROLL table UPDATE statement, the statement executed and the date and time.
3. In Oracle 9i, the ability to narrow the audit scope to detailed data driven conditions is not available, thus creating audit data overload, making it difficult for auditors to sift through all the records and determine which

transactions are questionable, unauthorized, or fraudulent. However, in 10g and beyond, "fine grained" auditing extends Oracle standard auditing capabilities by allowing the user to audit actions based on user-defined predicates.

4. SQL Audit has been known to impact system performance, so scope limitation is essential.
5. Although SQL Audit does come packaged with database views designed to facilitate reporting, it provides no end user reporting. Auditor end users may not consider those views easy to understand.
6. It is based upon SQL command line generated mechanisms which require creation and maintenance by a DBA or an equivalent technically minded resource.

Database Triggers

Database Triggers are another mechanism that is standard functionality in an Oracle database, as we have discussed above. They come in two types, table level and system level, and can audit both DML and DDL transactions. They are well suited for auditing and/or financial controls enforcement for the following reasons:

- They provide complete flexibility of audit scope and criteria. One can specify the precise condition upon which to audit a transaction.
- They provide record level and table column details, capturing the before and after impact on data of SQL statements, and any session level details desired.
- They provide a real time mechanism to create audit trails, **transmit email alerts**, and **prevent transactions**.
- They have access to useful Oracle *and* **E-Business Suite session details** of the transaction, like EBS Username and Responsibility, which can help identify the source of the transaction.
- They can capture **context/lookup** data from other associated tables (foreign key references) upon the triggering event, to incorporate the then current context in user-friendly form, so end user auditors can understand the audit record immediately without additional research.

Unfortunately, they too, have disadvantages, namely:

1. They are SQL command line generated mechanisms which require creation and maintenance by a DBA or equivalently technically minded resource with PL/SQL programming experience.
2. They are not secure from manipulation by a user with SYS access. The triggers themselves can be easily disabled or dropped, or the custom audit trail table may be deleted or changed.
3. DDL type triggers, also called system triggers, are configurable via the "_SYSTEM_TRIG_ENABLED" database initialization parameter. The default for this parameter is TRUE, but it may be disabled by a DBA after database shutdown.

4. A trigger, if not properly optimized and tested, can degrade its table's transaction performance.

On the other hand, **Absolute Technologies, Inc**. and other third party software vendors, consulting practices and customer project teams have eliminated some or all of the above-mentioned disadvantages and demonstrated that *the trigger approach is very effective*. These vendors offer packaged solutions ready for end users.

Log Miner

Audit initiatives can use Log Miner, another Oracle Database component. It was designed to provide information necessary for performing recovery operations. Essentially, Log Miner is a SQL command line mechanism which can review both online and archive redo log files of the database to extract the history of all transactions over a period of time for the given log file. It is well suited for auditing for the following reasons:

- Depending on your audit content requirements, it places little to no additional impact on database transactions, though it can require significant system resources and place an indirect load on transactions while a mining session is active.

- It can provide history for both DML and DDL transactions, starting in Oracle 9.2.

- It can provide before and after record level column details.

- Because it uses existing redo logs, it inherently has access to all transactions affecting the database. Therefore, one need not specify audit criteria until actively mining the logs. The audit criteria can be as wide or as narrow as the audit user desires. For example, all transactions made by a given user may be selected at once without specifying which objects were impacted. Alternatively, the audit user may select only those transactions affecting a given table, where a specified column was changed.

- If an organization retains the historical series of redo logs, it can use Log Miner to analyze the logs as part of the audit.

However, the disadvantages are:

1. Log Miner is a SQL command line based mechanism, which requires the expertise of a DBA or like-minded individual to maintain.

2. It is not secure from the DBA and users with SYS access, the very users, in this case, it is intended to audit.

3. Although Log Miner is equipped with views to access logged data, it provides no end user reporting.

4. To access logged transactions, one must initiate a Log Miner session from the SQL command line, make SQL style inquiries, then terminate the session when finished. The resulting output of the ad hoc inquiries is not saved. Therefore, to maintain an audit trail based upon desired audit criteria, a mechanism must be developed to load the results of such inquiries into a custom audit table to support future end user reporting and audit trail retention requirements.

5. Although Log Miner provides a transaction timestamp, it only ensures that transactions are sequenced in the order in which they were executed, making it an unreliable option to detect exactly when a change occurred.

6. Since redo log files only contain the data necessary to rebuild the database, they contain only a few of the session details available to triggers that could be used to identify the source of the transaction. For instance, values for terminal, module, program, application user, and responsibility, to name a few, are not available.

7. Unlike triggers, Log Miner lacks the ability to capture values from foreign key referenced tables at the time of transaction. By the time a Log Miner inquiry is made, the relevant data in the foreign key tables may have changed.

8. Log Miner is somewhat complex and tedious to use, resource intensive to maintain, and prone to administrator mistakes because each time it is initialized, the admin user must have and accurately specify each redo log file required for the given audit period.

9. Log Miner does not provide real time transaction auditing, but rather an after the fact view into the redo log files. For this reason, it does not support real time alerts or transaction prevention, though theoretically it could be used to "undo" an undesired or unauthorized transaction in certain circumstances.

10. It does not audit database user connections to the database.

The bottom line is that Log Miner is a good choice for ad hoc audit purposes when the database needs to be scrutinized for transactions meeting specific criteria during a specified timeframe. In order for it to be practical as on "always on", continuous mechanism to support compliance and control auditing and reporting requirements, however, it must be incorporated into a larger, more sophisticated, end user application. Fortunately, there are a few third party software vendors who have developed such applications that use Log Miner.

Fine Grained Auditing (FGA)

Fine Grained Auditing is a mechanism designed to audit SELECT access to tables at the record and column level. It is deployed and maintained by a DBA using packaged Oracle procedures from the SQL command line. This may be an important component of your audit strategy if you must maintain an audit trail of users who actually *view* sensitive or restricted data. However, it does require DBA level expertise to deploy and maintain, and has a **performance impact** that must be scrutinized and optimized.

AUDIT_SYS_OPERATIONS

In all of the above approaches to auditing the DBA, we have made it clear that they all embody a conflict of interest in that they generally *require a DBA to audit a DBA*. This may be less of an issue for organizations that have the resources to assign a distinct individual the segregated role of Security and Audit Administrator, whereby such an individual would be tasked to audit all database users and activity, DBAs included. Yet, even with such segregation of duties by such administrators, the necessity that one or any of the administrators have SYS access continues to present a conflict. It seems there will always be at least one individual with the "master key".

Fortunately, Oracle provides yet another mechanism to add a layer of security and auditability to the database with the initialization parameter called "AUDIT_SYS_OPERATIONS". The parameter enables all SYS activity in the database to be logged to the operating system, indiscriminately and regardless of the setting of the "AUDIT_TRAIL" initialization parameter.

For UNIX, the audit files are placed in the $ORACLE_HOME/rdbms/audit by default. However, the "AUDIT_FILE_DEST" initialization parameter may also be used to set an alternative directory based upon your operating system. To change the directory from the UNIX default, the oracle binary account must have write privileges to the directory specified, or Oracle will return an error at database startup.

Unfortunately, for UNIX, this directory is owned by the oracle binary account user, and thus is often accessible to DBAs. Additionally, given it's flat file structure, audit records generated using this method are not readily accessible or reportable by would be audit end users. Yet, an OS level mechanism owned by root may be deployed to synchronize, copy or move, the files to another directory owned by root and not accessible to the oracle binary user, thereby securing them. The utilities called Rsync or Unison may be of use in this effort.

Of course, the mention of the all-powerful OS "root" user begs the next security question, who audits root? That topic, however, is thankfully out of the scope of this whitepaper and will thus be left for others to address.

Recommendations

Now that the reader has covered all the basics, options and issues regarding the database, the DBA's role and dominion over it, and the different approaches to control and/or audit the DBA, we can provide a summary of recommendations directed at segregating the power of the DBA and generating a secure and systematic audit trail to prove it.

Segregate DBA Duties and Access

Given the powerful privileges of the DBA and his or her role in database auditing, it should now be clear that **a single all-powerful DBA** is a security *red flag*. If possible, the easiest way to lower the risk threshold is by creating at least two distinct, non-overlapping DBA positions in your organization. The first, we'll call the "Database and Application Support" DBA, who deals with all the normal database maintenance, trouble shooting, patching, upgrading and performance optimizing tasks and, the second, we'll call the "Security, Access and Auditing" DBA, who deals with granting system access to all users and invoking, maintaining and securing the audit trail mechanisms. In this way, the organization creates **a balance of powers** across the DBAs.

Although this step sets a good precedent in segregating the documented role of each DBA, there remains the technical challenge of allocating administration level access between the two roles. As we have pointed out earlier, the SYS user is all-powerful, so, at least one individual will retain that power. This opens a nice segue into our next recommendation.

Activate AUDIT_SYS_OPERATIONS

This database initialization parameter, when set to "TRUE", audits all activities performed by SYS and writes them to operating system log files in the default or user specified directory. Therefore, this step provides us with a check against the SYS user. However, this measure is only effective if the SYS user has no access to the audit files. This, in turn, brings us to our next step.

Protect the AUDIT_FILE_DEST Log Directory from the DBA

Depending on your operating system, you may be able to specify an audit directory that is inaccessible to the DBA via the oracle binary account or otherwise.

However, for UNIX servers, the audit directory must be accessible to Oracle, and thus the oracle binary account. Therefore, the SYS user, without even an OS account, can actually read, alter, and delete audit files from a SQL connection, if desired, given that SYS essentially connects to the OS as the oracle binary account. Alternatively, any user with access to the oracle binary account can access the audit files.

Therefore, in UNIX, this measure will not be sufficient alone. To protect the audit files, the OS root user must implement some OS level mechanism to copy, move, or synchronize the files to a root only accessible directory for safekeeping.

Limit Use of SYSDBA

In an earlier section, we warned that the above-mentioned approach to audit SYS did not provide an end user friendly mechanism for access and reporting. Nevertheless, it may be the only realistic and readily available approach, especially if it is complemented with a strict and enforced policy of limited access to the database as SYSDBA. In other words, if you seldom allow users to log onto the database as SYS, then you will generate minimal SYS audit transactions upon which to report and review. If such transactions begin to increase or become overwhelming, you know you have an access issue to resolve.

Limit OS User Assignment of the "DBA" Group

Do not fall into the trap of simply focusing on security at the database level. Remember that any OS user assigned to the DBA group has direct access to SYS, no database password needed. Include the restriction of this group in your high priority list of strictly enforced policies.

Utilize Named Accounts with the DBA Role

To improve visibility and accountability to DBA users, avoid providing shared access to the Oracle default accounts, both at the OS and database levels. Each individual connecting to the OS or the database should do so via their own, individually named account. This may add to user access maintenance, but it will greatly reduce the risk of fraud.

Specifically, instead of allowing DBAs to login to SYS as SYSDBA, provide them with a named account granted the DBA role. As a policy, query the database periodically to review and manage which users have been granted the DBA role.

```
SQL> select *
  2  from dba_role_privs
  3  where granted_role = 'DBA'
  4  /

GRANTEE                        GRANTED_ROLE                   ADM DEF
------------------------------ ------------------------------ --- ---
SYS                            DBA                            YES YES
SCOTT                          DBA                            NO  YES
CTXSYS                         DBA                            NO  YES
SYSTEM                         DBA                            YES YES
```

For database startup and shutdown operations, have the DBA connect as SYSOPER instead of SYSDBA. SYSOPER connects as the PUBLIC user, and has no access to the Data Dictionary or other schema's objects.

Define and Enforce a Password Security Policy

To tighten security across all default and named account users, systematic mechanisms should be deployed to ensure that passwords are changed periodically, are hard to guess, and are not reused with frequency.

Oracle provides user *Profiles* that may be created, assigned to users, and maintained by the security DBA to establish systematic password protection in the following areas:

- Account Locking
- Password Aging and Expiration
- Password History
- Password Complexity Verification

Audit DBA Activity on Key Application Objects

Deploy an audit trail mechanism that can provide detailed record and column level audit records, as well as the ability to define precise audit criteria by which to narrow the generated audit output to those transactions that have a greater probability of being unauthorized.

Both database triggers and Log Miner may be effectively used, separately or together, to achieve this. However, in either case, expect a significant development effort to create a maintainable, secure, and end user accessible solution. Alternatively, look to packaged software solutions which have already built a framework around these utilities to deliver a turn key database auditing solution.

For organizations running E-Business Suite, special audit attention may be warranted when transactions are performed using the APPS user via a "backend" connection to the database using a SQL tool like SQL*Plus or Toad verses a "front end" secure connection authenticated by E-Business Suite. As the reader may remember, E-Business Suite connects the application user to the database as APPS. The ability to distinguish between backend and front end connections using APPS can help identify unauthorized backend usage of the APPS user by DBAs or other privileged staff.

In this case, *triggers are the better audit mechanism* for accurately detecting the difference because they have access to the full range of session details during the transaction. They can more accurately identify if the APPS user is logged on via a front end connection as an application user, and determine the application user's name and responsibility. Log Miner fall short because it only provides limited session information, and does not provide application level user details.

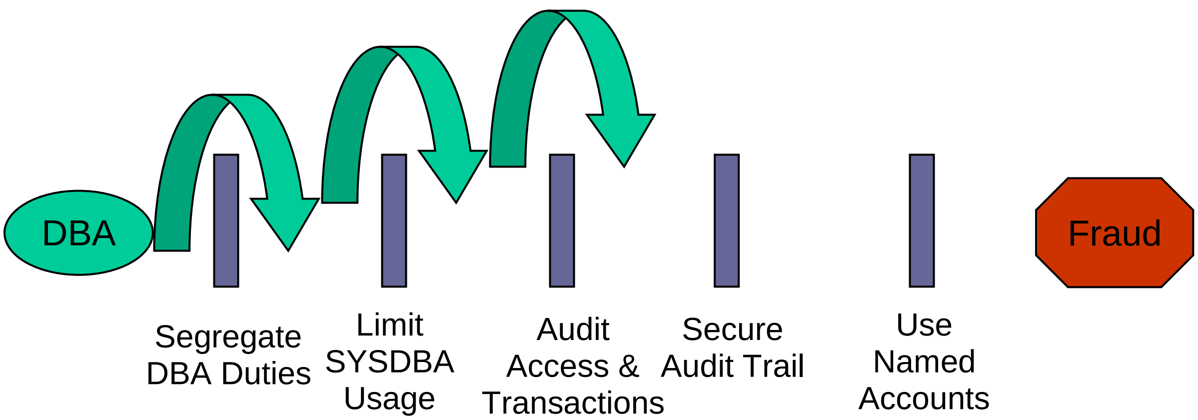
Monitor Key Initialization Parameters

Finally yet importantly, verify the values for key database initialization parameters discussed in this paper periodically and each time the database is restarted. Some parameters require that the database be shutdown and restarted to take effect, others can be modified while the database is up and running. Here is a simple SQL script which will get the job done.

```
select name, decode(type,1,'Yes','No') Require_Restart, value
from v$parameter
where name in (
'audit_sys_operations',
'audit_file_dest',
'audit_trail',
'remote_login_passwordfile',
'07_dictionary_accessibility',
'_system_trig_enabled')
/
```

Conclusion

Many may say because the DBA role is a trusted role and since a talented DBA can overcome almost any control and compliance restrictions, including an audit trail deployed to monitor DBAs, why try to control them? Whether DBAs can be controlled or not, the reality is that external auditors do scrutinize DBA access and request systematically proven controls before certifying compliance. Any particular approach may not be "bullet proof", but each hurdle or preventive measure deployed reduces the overall risk and will assist in achieving compliance in the eyes of auditors.



Hurdles to Mitigate Risk

References

- Oracle Database Administrator's Guide Release 2 (9 and 10g)
- Oracle New Features in Oracle Database Reference (9 and 10g)
- Oracle Advanced Security Administrator's Guide Release 2 (9 and 10g)
- Oracle SQL Reference Guide (9 and 10g)
- Oracle Database Reference (9 and 10g)
- Oracle Application Developer's Guide – Fundamentals (9 and 10g)
- Oracle Privacy Security Auditing *by Arup Nanda & Donald K. Burleson*

About the Author

Cameron Larner has over 25 years of extensive Oracle Applications experience. After earning a B.S. degree in Economics at the University of California at Berkeley in 1988, he joined a dynamic and rapidly growing Oracle Corporation, holding positions in Finance, Information Systems, and Applications Development.

In 1993, he left Oracle Corporation, to pursue an independent consulting role assisting companies implementing Oracle Applications, both domestically and internationally.

In 1997, he founded Absolute Technologies, Inc., an Oracle Applications solutions provider. As president of Absolute, he has developed and released several packaged software extensions for Oracle E-Business Suite, licensing them to over 30 customers. The first solution, BBB Intelligence, was first released in 1998. It provides Oracle Order Management users with critical, yet underdeveloped, reporting for Bookings, Backlog, and Billings transactions. The second solution, Application Auditor, was first released in 2005 to help customers comply with Sarbanes-Oxley regulations. It provides an automated approach to audit database transactions which may affect financial reporting or setups, and user assignment to application functionality which violates user defined Segregation of Duties controls. In 2006, Absolute released SOD Violations Manager to specifically and efficiently identify and resolve user access violations to segregation of duties compliance policy. In 2012, Absolute released ProVision Responsibilities and ProVision Resources, both designed to automate the request, approval and provisioning processes of critical system and non-system based resources to individuals, both employees and affiliates.

Mr. Larner continues to reside as president of Absolute, overseeing the design, development, support, marketing and sales of its software solutions. He has participated in user group activities throughout his career, and has written and presented several whitepapers on various subjects relating to Oracle Applications.

About Absolute Technologies [www.absolute-tech.com]

Absolute Technologies, Inc., helps companies increase user productivity through software extensions to Oracle E-Business Suite in the areas of GRC Auditing and Certification, Self Service Provisioning, and Revenue Visibility. Our innovative solutions dramatically simplify and automate critical business functions, reducing costs and time spent. We specialize in placing experts on the front line of support, so each customer gets fast and accurate resolution to their questions and issues.

Founded in 1997 by professionals from Oracle, our commitment to affordable, easily deployable solutions supported by experts continues to drive our customers' satisfaction and success. Our customers are diverse in size and industry, representing companies large and small.

Absolute Technologies is an OAUG member and Oracle Partner headquartered in Palo Alto, California.