ORACLE

Oracle Database Unified Audit: Best Practice Guidelines

A practical approach to auditing
Oracle Database activities

December, 2020, Version 21.1

Purpose statement

Oracle Database provides the industry's most comprehensive auditing capability, enabling the capture of detailed information relating to who, what, when the action was performed and the associated context with the activity which generated this audit record. This technical report provides an overview of auditing capabilities in Oracle Database, and helps you determine what to audit. It also helps you establish auditing baselines with best practice recommendations.

Executive summary

To create effective audit policies, we recommend auditing privileged user activity, security-relevant events, and sensitive data access. To build them, you can leverage a combination of several predefined audit policies and mandatory audit configurations provided by Oracle Database, apart from the recommended audit policies from Oracle Audit Vault and Database Firewall (AVDF) or Oracle Data Safe. You can also create customize your audit settings to monitor activities on sensitive application tables and wherever there is a need to incorporate policies unique to customer's scenario.

Sample audit configurations are provided in this technical report for illustrative purpose. For getting hands-on with the suggested audit policies, refer to Configuration of Sample Audit Policies in the appendix.

Intended audience

If you are responsible for designing, implementing, maintaining, or operating security controls for Oracle Databases, this paper is intended for you.

Disclaimer

Table of contents

Introduction

The organizations need cost-effective, easy-to-use security controls that can be quickly deployed across hundreds of databases to reduce risk and support regulatory compliance. These controls are divided into four broad categories:

1. **Assessment Controls** that examine the database to determine risk in the database's operation, configuration and content. These controls also evaluate the data within the database, identifying the types and amount of sensitive data.
2. **Detective Controls** that monitor user activities, and access to sensitive application data. This helps administrators detect threats so that they can be remediated. Detective controls are also frequently used to support compliance reporting, incident investigation, and resolution.
3. **Preventive Controls** that block unauthorized or out-of-policy access to data. Examples include encryption, redaction, masking, and restricting access to data by unauthorized users.
4. **Data-Driven Controls** that enforce fine-grained access within the database, providing a consistent authorization model across multiple applications, reporting tools, and database clients.

Database auditing is a key component of the Detective Controls. Auditing is the most effective way to record what happened in the database. Auditing tracks the use of privileges, activities of highly privileged users, access to sensitive data, actions performed on database objects and modifications made to database settings.

Database auditing has steadily increased in both capability and popularity over the past decade, and today is mandatory in most organizations. They need to audit not only to detect any unauthorized use, but also to ensure that they comply with different regulations, such as GDPR, PCI, CCPA and other privacy regulations across the globe.

Database auditing is typically used to:

- Monitor activities of privileged database administrators
- Detect unauthorized activity on sensitive assets
- Assist with investigations of data breaches or other suspicious activity
- Provide proof of monitoring critical assets to auditors
- Provide reports on changes to the database environment to auditors

Database auditing is the most accurate record of any database activity, not just from connections happening over the wire but also through direct local logins, recursive SQLs, dynamic SQLs, and stored procedures.

Oracle Database provides the industry's most comprehensive auditing capabilities providing detailed information. An audit record gives you full execution context including details of the operation, type of SQL statement executed, use of powerful system privileges, operation performed, database object involved in the operation, and other session details that are useful for forensic analysis.

Auditing can be configured to log both successful and unsuccessful operations, and include or exclude certain users from being audited. Auditing is independent of external connection factors like the network encryption, the access path, or the user, and is always available as a reliable source of actual events that have happened. Database auditing involves creating and enabling audit policies to track user actions, schema changes, logon events, etc.

Oracle Database auditing has been enhanced with each successive release of the database and today provides robust and highly customizable auditing that can be fine-tuned to address specific security requirements. For details on the major auditing functionality enhancements over successive releases, refer to the appendix section Auditing functionality by releases.

Database auditing is frequently augmented with Database Activity Monitoring (DAM) solutions that collect and store the audit data for alert generation, analysis, and reporting. Oracle Database security products that offer DAM solutions include Oracle Data Safe, and Oracle Audit Vault and Database Firewall (AVDF). These products provide a comprehensive approach, offering numerous benefits including:

- Consolidation of audit data in a dedicated repository that maintains the integrity and security of audit data
- Near real-time monitoring of all database activity and data security by first capturing audit data, and then analyzing and alerting on policy violations
- Providing out-of-the-box and customizable reports for security and compliance
- Ability to aggregate and search through audit records for forensic analysis

Reporting capabilities in Data Safe or AVDF provide detailed information on audit events collected from the source databases, and offer extensive filtering and analyzing capabilities to identify anomalous activity patterns such as the one shown below:

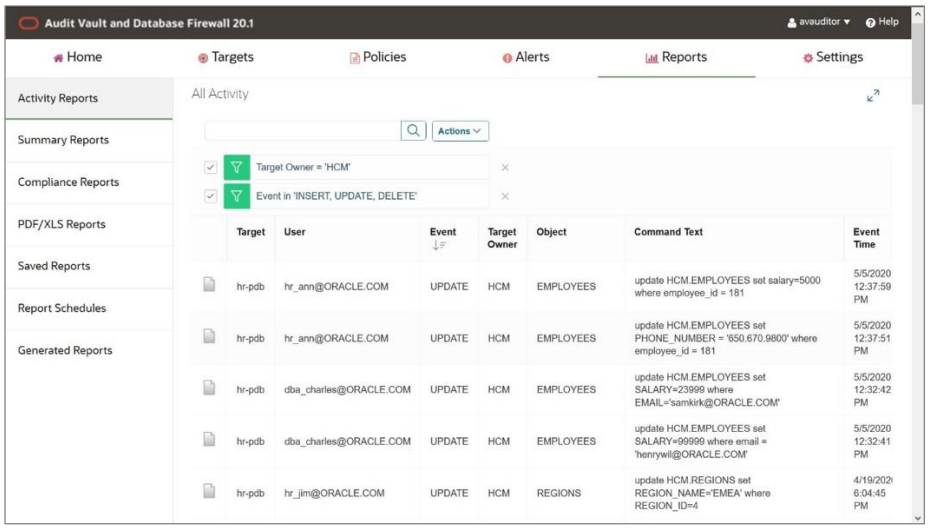


Figure 1: An AVDF report showing anomalous salary updates on sensitive employees table in HCM schema by certain privileged users

DAM solutions like AVDF can collect, consolidate and monitor audit data stream from heterogeneous systems including Oracle/non-Oracle databases, operating system audit trails, file systems, directory services, application audit data, SQL network traffic events and so on, thereby providing insight into attacks involving lateral movement. For example, there may be multiple failed login attempts occurring across multiple databases, indicating a possible brute force attack.

This technical report discusses the best practices of designing Oracle Database audit policies which are selective enough to reduce unnecessary audit records, but effective enough to provide a comprehensive view of database activity and lets you meet both security and regulatory compliance goals.

For a glossary of terms used in this technical report, refer to the appendix section Glossary of terms.

Introduction to Oracle Database auditing

Oracle Database provides robust audit support in all database editions, and provides a variety of information to detect any unauthorized use as shown below:

INFORMATION DETAILS	AUDIT EVENT DETAILS
Database details	• Database name • Database identifier • Instance number • Database link
Client details	• Host name • IP address • Terminal • Client Program • Global user • DB user • Proxy user • OS User • Client Identifier • Authentication type
Operation details	• Action • Success/Failure • Transaction identifier • Execution context identifier • Object name • Object owner • Object edition • System privileges • Object privileges • Timestamp (Local) • Timestamp (UTC) • Audit policies • Session Identifier
Statement details	• SQL statement • Bind values

Component details	• Fine-grained audit • Audit configuration changes • Virtual private database (VPD) • Oracle Database Vault • Oracle Label Security • Real Application Security • RMAN • Data Pump • Oracle SQL*Loader Direct Load • Oracle Kernel Access Control
Application details	• Application Context

Table 1: Audit event record details

Historically, **traditional auditing** in Oracle Database captured operations on privileges, objects, and statements. With privilege auditing, administrators can audit usage of a system privilege. With object auditing, administrators or the object's owner can audit DML commands, including queries against an object, and execution of PL/SQL procedures or functions. With statement auditing, administrators can audit selected DDL and DML commands. Audit records can be stored in the database audit trail or in the files on the operating system. To enable traditional auditing, a few initialization parameters had to be set to configure the audit trail, and AUDIT commands executed to identify which activity would be audited. Though still popular, traditional audit configurations are managed and configured individually, and lack the precision to capture only the activity that is relevant. Traditional auditing also has separate audit trails along with its custom format for different components like Database Vault, and Fine-Grained Auditing. Thus, consolidation and providing a comprehensive view of audit information across multiple traditional audit trails is a complex task, as is managing the audit configuration.

Unified audit was introduced in Oracle Database 12cR1, where auditing functionality of the Oracle Database underwent a significant redesign. Now, various audit trails have been unified into one audit trail and format, along with one unified audit policy simplifying its implementation. Unified audit further enables you to audit selectively by adding various conditions including application context values and simple built-in functions. This helps you to reduce the volume of your audit data, and at the same time helping you detect malicious activities in a timely manner.

With unified audit feature, audit configuration and management of audit trail are much simpler. Named audit policies can be created once and enforced in multiple dimensions (e.g. on users, roles). For instance, an audit policy definition to track all the top-level activities on a sensitive table like Employees in the HR schema if users are not using SSL authentication might look like:

```
CREATE AUDIT POLICY ALL_ACTIONS_ON_EMPLOYEES
       ACTIONS ALL ON HR.EMPLOYEES
       WHEN 'INSTR(UPPER(SYS_CONTEXT(''USERENV'', ''AUTHENTICATION_METHOD'')), ''SSL'') = 0'
       EVALUATE PER SESSION
       ONLY TOPLEVEL;
```

One can now enforce this audit policy on accounts granted DBA roles and track DBA activities on the Employee table:

```
AUDIT POLICY ALL_ACTIONS_ON_EMPLOYEES BY USERS WITH GRANTED ROLES DBA;
```

One can also enforce this audit policy on all accounts except trusted user 'HR_ANN':

```
AUDIT POLICY ALL_ACTIONS_ON_EMPLOYEES EXCEPT HR_ANN;
```

With unified audit, one can selectively audit to capture relevant activity. Note that the audit policy ALL_ACTIONS_ON_EMPLOYEES is defined to be precise and selective to track non-SSL sessions, which makes it easier to audit specific actions of interest and thus reduce the volume of irrelevant audit records. Audit conditions can be based on application contexts, session contexts, and built-in functions. The ONLY TOPLEVEL clause helps audit only the SQL statements that are directly issued by an end user, thus focusing only on end-user-initiated actions on sensitive table. Such configuration flexibility in unified audit helps fine-tune audit policies to collect audit data that is targeted to your needs.

Unified audit combines all database component audit trails into a single unified audit trail. Audit records are generated by a variety of audit sources including:

- Audit system related sources: audit records (including SYS audit records), mandatory audit records
- Security-control related sources: Oracle Database Vault, Oracle Label Security, Oracle Real Application Security
- Database operations related sources: Oracle Recovery Manager, Oracle Data Pump, Oracle SQL*Loader

With unified audit, audit records from all audit sources are written to a consolidated audit trail–AUDSYS.AUD$UNIFIED table or OS files, and exposed through the UNIFIED_AUDIT_TRAIL view. The unified audit trail also normalizes the audit record format, using standardized column names and data types across all audit sources. The consolidated, normalized unified audit trail simplifies collection, analysis, and management of audit records generated by different audit sources. Consistent formatting simplifies reporting and analysis of the audit data.

Unified audit offers high degree of integrity of audit trail by not allowing users to tamper with the audit trail. Unified audit trail is stored in AUDSYS schema and no one is allowed to login to that schema in the database. AUD$UNIFIED is a specialized table which allows only INSERT activity. Any attempt to directly truncate, delete or update contents of the AUD$UNIFIED table fail, and generate audit records. Audit data is managed using the built-in audit data management DBMS_AUDIT_MGMT package. Additionally, the audit tablespace can be encrypted with Transparent Data Encryption (TDE). The unified audit table can also be protected with a Database Vault realm.

The integrity of audit data can be verified as there can be two reliable sources of truth for unified audit data. With the UNIFIED_AUDIT_SYSTEMLOG parameter set, certain key fields of the unified audit records are written to syslog while the complete audit record is written to UNIFIED_AUDIT_TRAIL. As syslog records cannot be changed by the Oracle Database or its users, audit data in the unified audit trail can be verified with the audit fields from the syslog.

Unified audit trail can be extended to include application attributes by configuring auditing for application context values. Application context namespace can be populated with the required attributes, and those values are captured in the unified audit trail. For instance, the audit statement to audit the *clientcontext* application values for the module and service_name attributes might look like:

```
AUDIT CONTEXT NAMESPACE clientcontext ATTRIBUTES MODULE, SERVICE_NAME;
```

The APPLICATION_CONTEXTS column of the UNIFIED_AUDIT_TRAIL view is populated with the attribute values whenever an audit event is recorded as shown in the sample:

```
SELECT APPLICATION_CONTEXTS FROM UNIFIED_AUDIT_TRAIL;

APPLICATION_CONTEXTS

------------------------------------------------------------

(CLIENT_CONTEXT,MODULE=sqlplus@oracle.com (TNS V1-V3));  (CLIENT_CONTEXT,SERVICE_NAME=rac_service1.DEV)
```

Significant benefits of unified audit over traditional auditing are summarized in the appendix table Benefits of unified audit over traditional auditing.

Performance considerations of unified audit

For typical use cases of auditing privileged users or auditing key database operations, the performance impact is so low that it cannot even be measured due to low audit volume spread throughout the week. You could begin to see performance impact of 1% when the audit load increases to a few hundred audit events/second. For most use cases, you are not going to see overhead beyond this, but for cases where organizations want to audit application usage, it is best to tune the audit policies. Internal performance tests using a TPC-C mixed application workload show that with unified audit, you may see a CPU overhead in mid-single digit when auditing up to 360,000 audit records/hour. For extreme audit loads up to 1,800,000 audit records/hour, the additional overhead is still in a single digit.

As auditing is a transactional activity with typical ACID properties to guarantee record of database activities, we recommend that you fine-tune your audit policies to collect audit data that is targeted to your needs. Collecting unnecessary audit information impacts database performance, increases storage costs, and may make it more difficult to spot malicious database activity.

Customized audit with Fine Grained Auditing (FGA)

Oracle Database enables you to create customized audit policies using fine-grained auditing (FGA), which is available in Oracle Database Enterprise Edition. If the FGA policy specifies a column, then it will audit only the statements that reference the relevant column(s). With the help of the FGA, it becomes easier to focus on security-relevant columns such as national identifiers, birth dates, home addresses, and so on.

In general, FGA policies are based on simple, user-defined SQL predicates on table objects as conditions for selective auditing. During fetching, whenever policy conditions are met for a row, the query is audited. For instance, you can use FGA to audit granular actions such as:

- Access the sensitive data during non-business office hours
- Access the sensitive data from outside the corporate network
- Modify a sensitive data value above an expected threshold

Moreover, FGA allows you to monitor data access based on content of the column values returned. For instance, with FGA, you can audit access to a sensitive column like SALARY in the EMPLOYEES table only when record values with SALARY >1500 are retrieved by the query.

FGA policies also allow an event handler to be specified. Event handlers are PL/SQL functions called when an audit condition is triggered. When a SQL query satisfies the FGA policy conditions (i.e. relevant columns and specific data values being accessed), the event handler will be invoked which in turn can be configured to message a database administrator or trigger a security alert in an external system. This will speed up the detection of a security violation and allow administrators to respond to the problem sooner.

Two key use-cases where you will want to consider FGA policies in-addition to unified audit policies are:

1. When you want to audit access to specific security-relevant columns, and their sensitive data values
2. Raise alerts on possible security breaches

For instance, the sample FGA policy shown below tracks any updates to SALARY column values in EMPLOYEES table. Furthermore, the policy triggers an email alert by invoking a PL/SQL procedure EMAIL_ALERT owned by FGA_ADMIN user.

```
BEGIN
DBMS_FGA.ADD_POLICY(
       object_schema      => 'HR',
       object_name        => 'EMPLOYEES',
       policy_name        => 'updates_on_salary_column',
       audit_column       => 'SALARY',
       handler_schema     =>  'FGA_ADMIN',
       handler_module     =>  'EMAIL_ALERT',
       enable             =>  TRUE,
       statement_types    => 'UPDATE');
END;
```

Effective audit planning techniques

Effective database audit planning gives you the opportunity to streamline which database activities are monitored and tracked so that overall database performance impact is negligible, while not compromising on the detective requirements of the databases. Effectiveness of audit techniques can be regarded as a composite factor of multiple parameters, including:

1. Satisfying regulatory compliance needs
2. Adapting to the organization's risk tolerance, frequently driven by the type and quantity of sensitive data
3. Timely detecting the threat
4. Keeping audit within acceptable performance overhead limits
5. Effectively managing resulting audit data volume
6. Controlling storage costs required to house the collected audit data

While auditing every granular activity can potentially record inappropriate or malicious database activity, there is a trade-off on storage costs, performance impact, and efficiency. Effective auditing requires that audit policies capture the important details about significant auditable events, while reducing false positives and leading to quicker and reliable threat detection. It is essential to create the right audit policies to bridge the time gap between compromise and detection, i.e. shrinking detection time.

Create effective audit policies which are selective and targeted to your needs by focusing audit configuration on three factors–privileged user activity, security-relevant events, and sensitive data access.

Users, some of whom may be privileged, access databases. For example, database administrators (DBAs) are frequently considered **privileged users** because of their broad access within the database. Privileged user accounts are often soft targets for hackers attempting to gain access to critical systems and data. Continuous **privileged user activity monitoring** allows security teams to easily identify anomalous behavior and quickly detect sensitive data leaks.

Database users are usually granted privileges to perform operations within the database, and some of those privileges, such as SELECT ANY TABLE or ALTER USER, should be constantly monitored. Other noteworthy events in the database include failed login attempts, schema structural changes, privilege grants, and so on. Such actions within the database warrant greater scrutiny and constant monitoring because they can potentially be abused and are categorized into **security-relevant events**. Monitoring such actions constitutes **security-relevant events auditing** and helps detect anomalous activities in the database.

Databases usually contain **sensitive data**–data whose access should be controlled and monitored. Examples of sensitive data might include financial data, credit card numbers, email addresses, and other personal data that describes an employee or customer. **Sensitive data access auditing** is a powerful monitoring mechanism providing visibility into access and changes to sensitive data and may serve as a primary deterrence to those who do not have a business reason to access or modify them.

Focusing audit configuration on **privileged user activity, security-relevant events, and sensitive data access** helps build better audit policies–polices that are focused on the activities that matter, selective enough to reduce the creation of unnecessary audit records, and effective enough to let you meet your audit goals.

Let us examine in detail the audit policies configuration for these use cases:

- Privileged user activity auditing
- Security-relevant events auditing
- Sensitive data access auditing

Certain security sensitive database activities are always audited in Oracle Database and cannot be disabled. They are listed in the appendix section Mandatory audit configurations of Oracle Database. One can also leverage several predefined "best practice" unified audit policies that cover common security-relevant audit settings as detailed in the appendix section Predefined unified audit policies of Oracle Database. If you are using Data Safe or AVDF to monitor database activity, you can provision many of the recommended audit configurations in addition to provisioning predefined policies of Oracle Database. For details on provisioning options, refer to the section Summary of audit policies and provisioning.

Many of the sample audit configurations covered in this section are provided as SQL scripts, which can be executed in the HR demo schema. Refer to the appendix section Configuration of sample audit policies for details on how you can configure them in the HR demo schema and execute the workload and generate the audit events.

Privileged user activity auditing

Certain database user accounts are privileged due to their job responsibility to work directly with infrastructure. Such accounts typically have database administration privileges enabling direct access to all data including sensitive data. Database administrators also have the capability to manage database user accounts, export data, and modify configuration. Accounts with such wide access are a popular target for cyber criminals and pose a serious threat if compromised.

Full auditing of privileged user activity allows security teams to detect inappropriate activity in a timely fashion. Different types of privileged users whose activity needs to closely audited are shown below:

AUDIT FOR PRIVILEGED USER ACTIVITY
• Audit administrative database user accounts
• Audit database user accounts with direct access
• Audit individual high risk database user accounts

Table 2: Audit for privileged user activity

Let us examine in detail the audit policies needed for each of the above privilege user types.

Audit administrative database user accounts

First, identify database user accounts with administrative privileges and then configure audit policy for such accounts.

You can identify database user accounts with administrative privileges from different sources like User Assessment report in Data Safe such as the one shown below, or from DBSAT report:

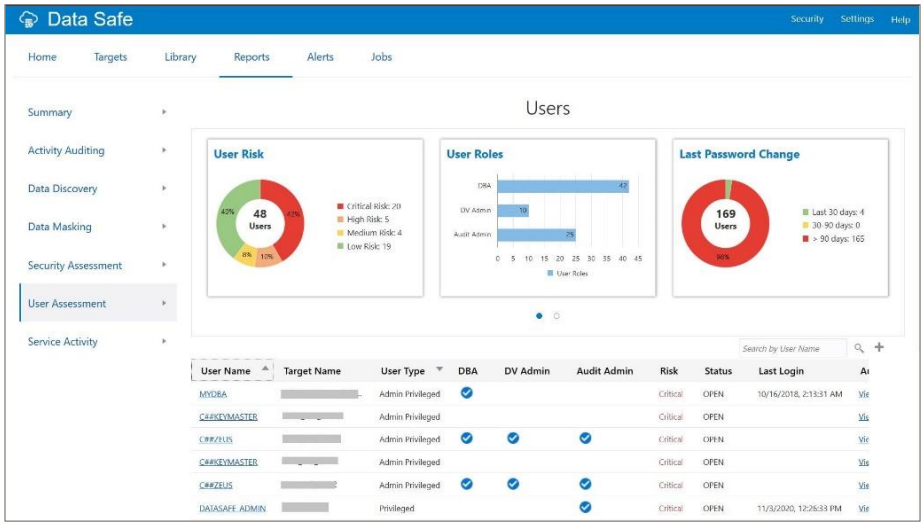


Figure 2: User Assessment report in Data Safe showing Privileged and Admin Privileged users

Once you identify the database user accounts with administrative privileges, configure audit policy for such accounts. Top-level statements by administrative users (e.g. SYSDBA, SYSKM) are mandatorily audited when the database is in the closed or mount state. Most organizations want to also audit activity by these administrative users once the database is open. Create audit policies to capture all top-level actions of administrative user accounts, such as SYS during normal database operations. Another common organization requirement is to audit activity by database administrators (e.g. SYSTEM, PDB_ADMIN, or named DBA accounts).

If you are using Data Safe or AVDF to monitor database activity, you can provision recommended audit policy "Admin Activity Auditing" which lets you audit all activities by privileged administrators, including SYS. They provision the following functionally similar audit policies:

MONITORING MECHANISM	RECOMMENDED AUDIT POLICY
Data Safe	Admin Activity Auditing policy (`ORA_ADS$_ADMIN_USER_ACTIVITY, ORA_ADS$_SYS_TOP_ACTIVITY`)
AVDF	Admin Activity Auditing policy (`ORA_AV$_ADMIN_USER_ACTIVITY, ORA_AV$_SYS_TOP_ACTIVITY`)

Table 3: Audit policy for administrative database user accounts

If you are not using Data Safe or AVDF and want to create equivalent audit policies, create an audit policy such as the one shown below:

```
CREATE AUDIT POLICY ALL_ACTIONS_BY_PRIVILEGED_USERS
       ACTIONS ALL ONLY TOPLEVEL;
AUDIT POLICY ALL_ACTIONS_BY_PRIVILEGED_USERS BY SYS, SYSKM, SYSBACKUP, SYSRAC, SYSDG, PUBLIC;
AUDIT POLICY ALL_ACTIONS_BY_PRIVILEGED_USERS BY USERS WITH GRANTED ROLES DBA;
AUDIT POLICY ALL_ACTIONS_BY_PRIVILEGED_USERS BY MYDBA;
```

Note: As shown in the sample, enforce the audit policies for privileged database user accounts including:

1. *Predefined Oracle accounts with administrative privileges (SYS, SYSKM, SYSBACKUP, SYSRAC, SYSDG, and so on)*
2. *Accounts granted the DBA roles.*

3. *Database users with administrative privileges as shown in User Assessment report in Figure 3 (MYDBA, and so on)*

Notice that for DBA activity, we audit the use of any of the system privileges that are part of the DBA role, not the user granted the role. This helps mitigate the risk that a DBA could create another DBA user and perform activity as that user which might not be audited if we audited DBA activity by explicit username. It is good practice to not use the DBA role, but rather create your own custom role with relevant privileges, and then use that role in your audit policies.

Also auditing just the role means that we do not audit statements that did not require use of that role, and thus avoid the need to create unneeded audit records.

Audit database user accounts with direct access

Outside of Database Administrators, direct access to databases is usually granted to only power-users like data analysts or application administrators, but not application end-users. Direct access to the database may come through a remote connection or through local sessions on the database server initiated through SSH/VNC/RDP.

Auditing local sessions is crucial because they bypass network monitoring. We recommend auditing of all top-level operations originating from a local session.

Local direct access to Oracle Database typically has IP_ADDRESS in USERENV set to null. The audit condition with the below filtering, will help identify the local direct access (including bequeath connections):

SYS_CONTEXT('USERENV','IP_ADDRESS') IS NULL

Create audit policy to audit all top-level actions as shown below:

```
CREATE AUDIT POLICY DIRECT_DB_ACCESS
       ACTIONS ALL
       WHEN '(SYS_CONTEXT (''USERENV'', ''IP_ADDRESS'') IS NULL)'
       EVALUATE PER SESSION
       ONLY TOPLEVEL;

AUDIT POLICY DIRECT_DB_ACCESS;
```

Note: If there are application service accounts that are logging into the database directly without connecting via middle-tier application (i.e. outside of trusted application paths), the audit policy when enforced on all users helps tracks such access.

Data access outside the application is important to audit because it is ad-hoc and not constrained by pre-defined application capabilities. It is relatively easy to detect such unusual activity with auditing.

Audit individual high risk database user accounts

In some cases, it will be necessary to closely monitor all user-initiated activities of individual database accounts–this may be because these individuals are higher risk, or because they have access to sensitive data.

If you are using Data Safe or AVDF, you can provision the audit policy “User Activity Auditing” which lets you audit all activities by the given set of users. They provision the following functionally similar policies:

MONITORING MECHANISM	RECOMMENDED AUDIT POLICY
Data Safe	User Activity Auditing policy (`ORA_ADS$_USER_ACTIVITY`)
AVDF	User Activity Auditing policy (`ORA_AV$_USER_ACTIVITY`)

Table 4: Audit policy for individual database user accounts

If you are not using Data Safe or AVDF, consider creating an audit policy like the one shown here:

```
CREATE AUDIT POLICY ALL_ACTIONS_BY_NAMED_USERS
       ACTIONS ALL ONLY TOPLEVEL;
AUDIT POLICY ALL_ACTIONS_BY_NAMED_USERS BY jron, lucas;
```

Here "jron" and "lucas" are the users provided direct access to the database. In addition to explicitly naming users, you can also audit all users EXCEPT a given set of users.

Security-relevant events auditing

Some database actions need to be monitored closely because they have broader impact than just one table or schema, and could be indicative of potential abuse, or hiding of events. Such actions include:

AUDIT FOR SECURITY-RELEVANT EVENTS
• Audit security-management events
• Audit account-management events
• Audit data-security events
• Audit database-management events
• Audit data-management events
• Audit activities with system privileges
• Audit activities of unused system privileges
• Audit usage of components with data implications
• Monitor suspicious user-activity events

Table 5: Audit for tracking security-relevant events

Let us examine the audit policy configuration for each of the above scenarios.

Audit security-management events

We must audit any changes in the database-wide security policies such as the following:

SECURITY-MANAGEMENT EVENTS	TYPICAL USER COMMANDS
Security parameters changes such as `SEC_MAX_FAILED_LOGIN_ATTEMPTS`, `SELECT_ANY_DICTIONARY`, etc.	ALTER DATABASE, ALTER SYSTEM
Audit policies changes	ALTER AUDIT POLICY
Key rotation	ADMINISTER KEY MANAGEMENT

Table 6: Typical security-management events

If you are using Data Safe or AVDF, you can provision the recommended audit policy "Critical Database Activity". They create the following functionally similar policies:

MONITORING MECHANISM	RECOMMENDED AUDIT POLICY
Data Safe	Critical Database Activity policy (`ORA_ADS$_CRITICAL_DB_ACTIVITY`)

AVDF	Critical Database Activity policy (`ORA_AV$_CRITICAL_DB_ACTIVITY`)

Table 7: Audit policy for security-management events

If you are not using Data Safe or AVDF, enable at a minimum the pre-defined audit policies of Oracle Database–ORA_SECURECONFIG and ORA_ACCOUNT_MGMT highlighted in the appendix section Predefined unified audit policies of Oracle Database. Any changes in audit policies are mandatorily audited as highlighted in the appendix section Mandatory audit configurations of Oracle Database.

Audit account-management events

Account-management events are related to users, roles, privileges, grants, revokes, etc., including BECOME_USER. These events need to be closely scrutinized as you are altering who has access to the database.

ACCOUNT-MANAGEMENT EVENTS	TYPICAL USER COMMANDS
User security-profile changes	CREATE/ALTER/DELETE USER/ROLE/PROFILE GRANT/REVOKE

Table 8: Typical account-management events

If you are using Data Safe or AVDF, you can provision the recommended audit policy "Critical Database Activity" as listed in the Table 7: Audit policy for security-management events.

If you are not using Data Safe or AVDF, enable at a minimum the pre-defined audit policies of Oracle Database–ORA_SECURECONFIG and ORA_ACCOUNT_MGMT highlighted in the appendix section Predefined unified audit policies of Oracle Database.

Audit data-security events

Data-security events typically track any security policy changes around protected objects or schemas. Typical data-security events and their respective audit policy configurations are highlighted in the table here.

DATA-SECURITY EVENTS	TYPICAL USER COMMANDS	AUDIT POLICY RECOMMENDATION
Redaction policy changes	DBMS_REDACT procedures	`CREATE AUDIT POLICY redaction_policy_changes ACTIONS EXECUTE ON DBMS_REDACT;`

VPD/OLS/RAS policy changes	DBMS_RLS procedures	**For VPD:**
		`CREATE AUDIT POLICY vpd_policy_changes ACTIONS EXECUTE ON DBMS_RLS;`
		
		For OLS:
		`CREATE AUDIT POLICY ols_policy_changes ACTIONS COMPONENT=OLS ALL;`
		
		For RAS:
		`ORA_RAS_POLICY_MGMT and ORA_RAS_SESSION_MGMT`

Database Vault policy changes	DBMS_MACADM procedures	DV admin tasks including all policy changes are mandatorily audited as listed in the appendix section Mandatory audit configurations of Oracle Database.
TSDP policies changes	DBMS_TSDP_MANAGE / DBMS_TSDP_PROTECT procedures	`CREATE AUDIT POLICY tsdp_policy_changes ACTIONS EXECUTE ON DBMS_TSDP_MANAGE, EXECUTE ON DBMS_TSDP_PROTECT;`

Exempt policies changes	EXEMPT ACCESS POLICY	Predefined Oracle Database policy `ORA_SECURECONFIG tracks EXEMPT ACCESS POLICY and EXEMPT REDACTION POLICY.`

Table 9: Audit policy for data-security events

Audit database-management events

Typical database-management events of relevance for auditing are highlighted in the table here.

DATABASE-MANAGEMENT EVENTS	TYPICAL USER COMMANDS	AUDIT CONFIGURATION
Backup/restore operations	RMAN operations	Oracle Recovery Manager events are part of mandatory auditing in the Oracle database as listed in the appendix Mandatory audit configurations of Oracle Database.
Cloning PDBs	RMAN Duplication command	
Creating/deleting tablespace	CREATE/ALTER/DROP TABLESPACE	CREATE AUDIT POLICY tablespace_changes ACTIONS create tablespace, alter tablespace, drop tablespace;
Patching	opatch	If database is mounted, SYS actions should be audited using best practices suggested in section Audit administrative database user accounts. If database is unmounted during patch application, SYS activities are captured in mandatory auditing of Oracle database.
Altering Database	ALTER DATABASE, ALTER SYSTEM	Predefined Oracle Database policy ORA_SECURECONFIG
Create/alter/delete package/function/synonyms/library		Consider the audit policy configuration as given in Section Audit data-management events.

Table 10: Audit policy for database-management events

Audit data-management events

It is crucial to track database schema structure modification events like create/alter/delete of tables/index/views, for all users. If you are using Data Safe or AVDF, you can provision recommended audit policy “Database Schema Changes”:

MONITORING MECHANISM	RECOMMENDED AUDIT POLICY
Data Safe	Database Schema Changes policy (`ORA_ADS$_DB_SCHEMA_CHANGES`)
AVDF	Database Schema Changes policy (`ORA_AV$_DB_SCHEMA_CHANGES`)

Table 11: Audit policy for data-management events

If you are not using Data Safe or AVDF, and want to create a similar audit policy, consider creating the audit policy shown here:

```
CREATE AUDIT POLICY AUDIT_DB_SCHEMA_CHANGES
PRIVILEGES
       CREATE EXTERNAL JOB, CREATE JOB, CREATE ANY JOB
ACTIONS
       CREATE PACKAGE, ALTER PACKAGE, DROP PACKAGE,
       CREATE PACKAGE BODY, ALTER PACKAGE BODY, DROP PACKAGE BODY,
       CREATE PROCEDURE, DROP PROCEDURE, ALTER PROCEDURE,
       CREATE FUNCTION, DROP FUNCTION, ALTER FUNCTION,
       CREATE TRIGGER, ALTER TRIGGER, DROP TRIGGER,
       CREATE LIBRARY, ALTER LIBRARY, DROP LIBRARY,
       CREATE SYNONYM, DROP SYNONYM, ALTER SYNONYM,
       CREATE TABLE, ALTER TABLE, DROP TABLE, TRUNCATE TABLE,
       CREATE DATABASE LINK, ALTER DATABASE LINK, DROP DATABASE LINK,
       CREATE INDEX, ALTER INDEX, DROP INDEX,
       CREATE INDEXTYPE, ALTER INDEXTYPE, DROP INDEXTYPE,
       CREATE OUTLINE, ALTER OUTLINE, DROP OUTLINE,
       CREATE CONTEXT, DROP CONTEXT,
       CREATE ATTRIBUTE DIMENSION, ALTER ATTRIBUTE DIMENSION, DROP ATTRIBUTE DIMENSION,
       CREATE DIMENSION, ALTER DIMENSION, DROP DIMENSION,
       CREATE MINING MODEL, ALTER MINING MODEL, DROP MINING MODEL,
       CREATE OPERATOR, ALTER OPERATOR, DROP OPERATOR,
       CREATE JAVA, ALTER JAVA, DROP JAVA,
       CREATE TYPE BODY, ALTER TYPE BODY, DROP TYPE BODY,
       CREATE TYPE, ALTER TYPE, DROP TYPE,
       CREATE VIEW, ALTER VIEW, DROP VIEW,
       CREATE MATERIALIZED VIEW, ALTER MATERIALIZED VIEW, DROP MATERIALIZED VIEW,
       CREATE MATERIALIZED VIEW LOG, ALTER MATERIALIZED VIEW LOG, DROP MATERIALIZED VIEW LOG,
       CREATE MATERIALIZED ZONEMAP, ALTER MATERIALIZED ZONEMAP, DROP MATERIALIZED ZONEMAP,
       CREATE ANALYTIC VIEW, ALTER ANALYTIC VIEW, DROP ANALYTIC VIEW,
       CREATE SEQUENCE, ALTER SEQUENCE, DROP SEQUENCE,
       CREATE CLUSTER, ALTER CLUSTER, DROP CLUSTER, TRUNCATE CLUSTER;
AUDIT POLICY AUDIT_DB_SCHEMA_CHANGES;
```

Audit activities with system privileges

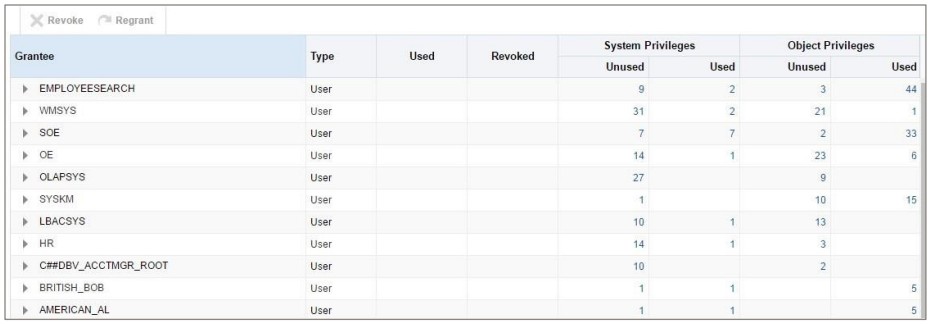
When someone is granted database privileges that exceed the requirements of their job function, these privileges can be abused. Sometimes, administrators grant excessive system privileges just to avoid the risk of failures due to lack of access privileges, or users may simply accumulate such privileges over time. System privileges are very powerful as they allow access to objects across multiple schemas or allow you to make changes that impacts the entire database. Such privileges should be granted only when necessary, preferably to roles and trusted users of the database. Use of system privileges should be monitored very closely.

The first step is to identify the system privileges granted to the database users that are currently in-use. Then configure audit policies to track the activities that makes use of the privileges.

Identify system privileges and their grantee information using Privilege Analysis (PA)–a feature of Oracle Database Enterprise Edition. PA dynamically analyzes privilege and role usage for database users and application service accounts. PA generates reports on which roles/privileges were used as well as those granted roles/privileges that

were not used. Understanding actual usage of roles and privilege is essential to implementing a least privilege model for all database accounts and reducing your application attack surface.

Sample page in Enterprise Manager showing the system privileges granted to users, along with usage data:



Grantee	Type	Used	Revoked	System Privileges		Object Privileges	
				Unused	Used	Unused	Used
EMPLOYEESEARCH	User			9	2	3	44
WMSYS	User			31	2	21	1
SOE	User			7	7	2	33
OE	User			14	1	23	6
OLAPSYS	User			27		9	
SYSKM	User			1		10	15
LBACSYS	User			10	1	13	
HR	User			14	1	3	
C##DBV_ACCTMGR_ROOT	User			10		2	
BRITISH_BOB	User			1	1		5
AMERICAN_AL	User			1	1		5

Figure 3: System privileges landscape (unused/used privileges)

Optionally, one can query the Privilege Analysis view: DBA_USED_SYSPRIVS that is populated after the capture process. Refer to the section “Performing Privilege Analysis to Find Privilege Use” in Oracle Database Security guide for more details on configuration of Privilege Analysis.

Once you identify the system privileges granted to database users that are currently in-use, configure audit policies to track the activities that makes use of the granted privileges.

Remember that system privileges are very powerful and should only be granted when necessary to roles and trusted users of the database. Consider maintaining a runbook of trusted users and granted system privileges. If you are using AVDF, you can also configure baseline entitlement snapshots in place of the runbook and see how privileges have changed over time.

Create policies that audit activities using a system privilege. To learn more about system privileges auditing, refer to the section “Auditing System Privileges” in Oracle Database Security Guide. Here is a sample audit policy tracking the use of certain system privileges for all users:

```
CREATE AUDIT POLICY ALL_ACTIONS_USING_SYSTEM_PRIV
PRIVILEGES
       SELECT ANY TABLE, UPDATE ANY TABLE, INSERT ANY TABLE, REDEFINE ANY TABLE, DELETE ANY TABLE,
       ALTER ANY ROLE, ALTER ANY TRIGGER, DROP ANY ROLE,
       CREATE ANY CONTEXT, DROP ANY CONTEXT,
       GRANT ANY PRIVILEGE, GRANT ANY ROLE, GRANT ANY OBJECT PRIVILEGE;
       -- Include all the used system privileges from the Privilege Analysis (PA) report
AUDIT POLICY ALL_ACTIONS_USING_SYSTEM_PRIV;
```

Note: You can exclude UNLIMITED TABLESPACE system privilege from the audit policy configuration even if it is in used list of system privileges.

As the system privilege entitlements could vary over time, consider automating the task by creating a periodic job that executes the privilege analysis capture process and lists the used system privileges. Within the periodic job, you can disable/drop the audit policy and create/enable the audit policy for the latest set of used system privileges.

Audit activities of unused system privileges

Closely monitor unused system privileges to confirm if they can be safely revoked from users. It is recommended to reduce the attack surface and increase operational security by identifying and revoking the granted but unused

system privileges, if operations are not impacted. In many cases it is prudent to wait an extended time before revoking privileges to ensure they are not simply used infrequently (e.g. end of year processing). In this case, configure audit policies to track the use of those privileges until you are certain they can be safely revoked.

The first step is to identify the system privileges granted to the database users that are not being used, then identify the database users authorized to use them. Privilege analysis helps identify the system privileges that are granted to the database users and are not being used, as shown in Figure 3: System privileges landscape (unused/used privileges).

Optionally, query the privilege analysis view: DBA_UNUSED_SYSPRIVS that is populated post the capture process. Refer to the section 'Performing Privilege Analysis to Find Privilege Use' in the Oracle Database Security guide for more details on using privilege analysis to identify unnecessary privilege and role grants.

As covered in the previous section, consider maintaining a runbook of trusted users and granted system privilege in the environment. If you are using AVDF, configure baseline entitlement snapshots as the runbook.

Here is a sample audit policy where the database user "secadmin_steve" is the authorized database user who was granted the set of system privileges, which do not appear to be in use. If these do get used, they will create audit records.

```
CREATE AUDIT POLICY ALL_ACTIONS_UNUSED_SYSTEM_PRIV
       PRIVILEGES ALTER ANY ROLE, ALTER ANY TRIGGER, DROP ANY ROLE, GRANT ANY PRIVILEGE;
       -- Include all the unused system privileges from the Privilege Analysis (PA) report that need to exist

AUDIT POLICY ALL_ACTIONS_UNUSED_SYSTEM_PRIV BY secadmin_steve;
```

Audit usage of components with data implications

Consider auditing of database components such as Oracle Database Vault, Oracle Data Pump, Oracle SQL*Loader, Oracle Label Security, and Oracle Database Real Application Security, when these are being used.

A sample audit policy to monitor Data Pump operations might look like:

```
CREATE AUDIT POLICY AUDIT_DATAPUMP
       ACTIONS COMPONENT= datapump EXPORT, IMPORT;

AUDIT POLICY AUDIT_DATAPUMP ;
```

If Oracle Database Real Application Security is enabled, use predefined unified audit policy ORA_RAS_POLICY_MGMT.

If Oracle Database Vault is enabled, use predefined unified audit policies ORA_DV_AUDPOL and ORA_DV_AUDPOL2.

Monitor suspicious user-activity events

Suspicious database activity can refer to a number of different behaviors that seem unusual or out of place, like abnormal access patterns, or any out-of-the-ordinary database actions that can indicate an attack or data breach. Being able to recognize these activities is important as it can help pinpoint the source, allowing you to act quickly to correct the security threat and minimize damage. Let us examine common abnormal access patterns indicating suspicious activity:

- Multiple failed login attempts
- Sudden activity in dormant accounts
- Non-business hour activities

Multiple failed login attempts

Multiple, consecutive failed authentication attempts over a short period indicates that an account is under a brute-force attack.

If you are using Data Safe or AVDF, provision recommended audit policy "Login Events" which creates audit events for all successful or failed login/logout events. This is very useful if you want to find out who logged in, when, and from where to help you in analyzing incidents.

MONITORING MECHANISM	RECOMMENDED AUDIT POLICY
Data Safe	Login Events policy (ORA_ADS$_LOGON_EVENTS, ORA_ADS$_LOGON_FAILURES)
AVDF	Login Events policy (ORA_AV$_LOGON_EVENTS, ORA_AV$_LOGON_FAILURE)

Table 12: Audit policy for multiple failed login attempts

If you are not using Data Safe or AVDF, enable at a minimum the pre-defined audit policy of Oracle Database–ORA_LOGON_FAILURES highlighted in the appendix section Predefined unified audit policies of Oracle Database.

Sudden activity in dormant accounts

While monitoring active and privileged access accounts is crucial, keeping track of inactive or dormant users is also a critical security requirement. Inactive or expired temporary or dormant accounts can expose organizations to a range of privilege escalation threats and data breaches. Tracking when database users last logged in is a common security and compliance requirement–for example to reconcile users who are no longer in the system, have changed jobs, and to track any malicious activity from these dormant database accounts.

The first step is to identify the dormant database user accounts, and then configure audit policies to track all top-level activities of dormant accounts. DBSAT helps identify dormant database user accounts as shown here:

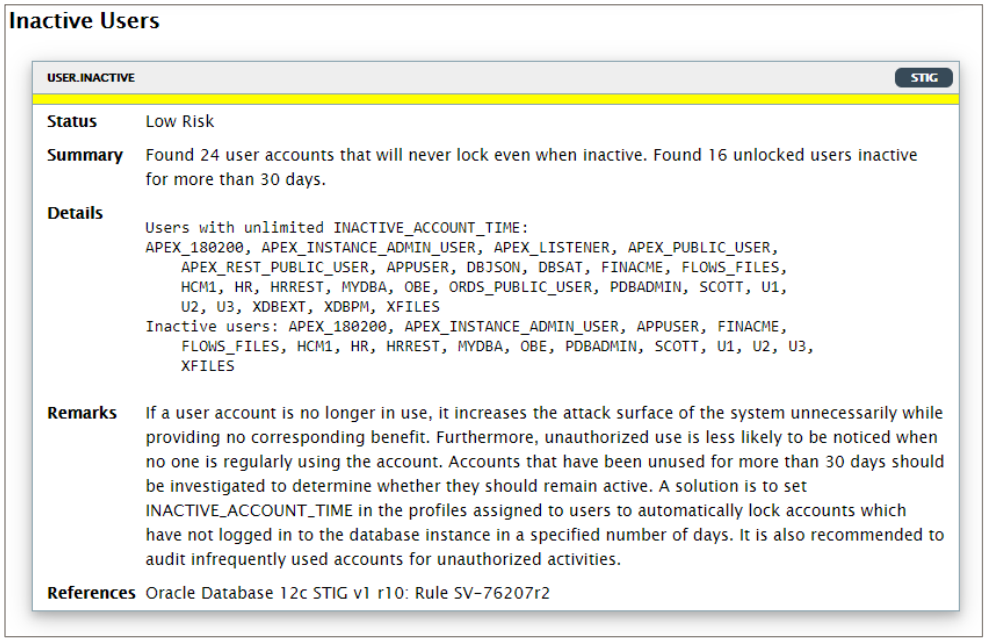


Figure 4: Dormant user accounts in DBSAT report

Data Safe also shows data regarding inactive database user accounts.

Once you identify the dormant database user accounts, configure audit policies to track activities of such accounts. A sample audit policy might look like:

```
CREATE AUDIT POLICY ALL_ACTIONS_BY_DORMANT_USERS
       ACTIONS ALL
       ONLY TOPLEVEL;

AUDIT POLICY ALL_ACTIONS_BY_DORMANT_USERS BY APPUSER;
```

Where APPUSER is one of the dormant users reported by Data Safe or DBSAT. Enforce the audit policy for the list of dormant users as retrieved from the DBSAT report.

Consider automating the task of identifying dormant database user accounts by executing a job that runs a script, which queries DBA_USERS for LAST_LOGIN column value. Based on the corporate requirement, determine the number of days that would identify an account to be dormant. Enforce the audit policy for the newer set of dormant accounts within the job.

You can also track dormant database user activity in AVDF summary reports as shown below:

Figure 5: AVDF Dormant user activity report showing sudden surge in activity of dormant user SHRI who has been inactive otherwise for months

Non-business hour activities

Create audit policy to track all actions of users during non-business hours. A sample audit policy might look like:

```
CREATE AUDIT POLICY AUDIT_NON_BUSINESS_HOURS
       ACTIONS update ON HR.EMPLOYEES
       WHEN '((SYS_CONTEXT(''DATE_CTX'',''DAY'') NOT IN ''SATURDAY, SUNDAY'')
       AND (SYS_CONTEXT(''DATE_CTX'',''TIME'') > ''180000'')
       AND (SYS_CONTEXT(''DATE_CTX'',''TIME'') < ''090000'')) OR
       (SYS_CONTEXT(''DATE_CTX'',''DAY'') IN ''SATURDAY, SUNDAY'')'
       EVALUATE PER STATEMENT;

AUDIT POLICY AUDIT_NON_BUSINESS_HOURS EXCEPT HR_ANN;
```

Where DATE_CTX represents user application context representing the date and is populated in the logon trigger.

Note: *The audit policy is enforced on all users except "HR_ANN" who is authorized to work after office hours.*

Sensitive data access auditing

Selective auditing of sensitive data access provides an effective way to monitor access and updates to sensitive data. Prior to configuring audit policies for sensitive data access, you need to know what sensitive information is contained within the database, which database tables contain the sensitive data, and who can access the data.

The first step is to identify and understand your sensitive data landscape. Data sensitivity is generally context sensitive. It depends on many factors, including regulations, company policy, contractual obligations, and user expectations.

There are several sensitive data discovery tools available to identify and categorize sensitive data, including Data Safe, DBSAT, and Enterprise Manager.

Here is a sample Data Safe's easy-to-use sensitive data discovery feature report to identify sensitive tables and columns:

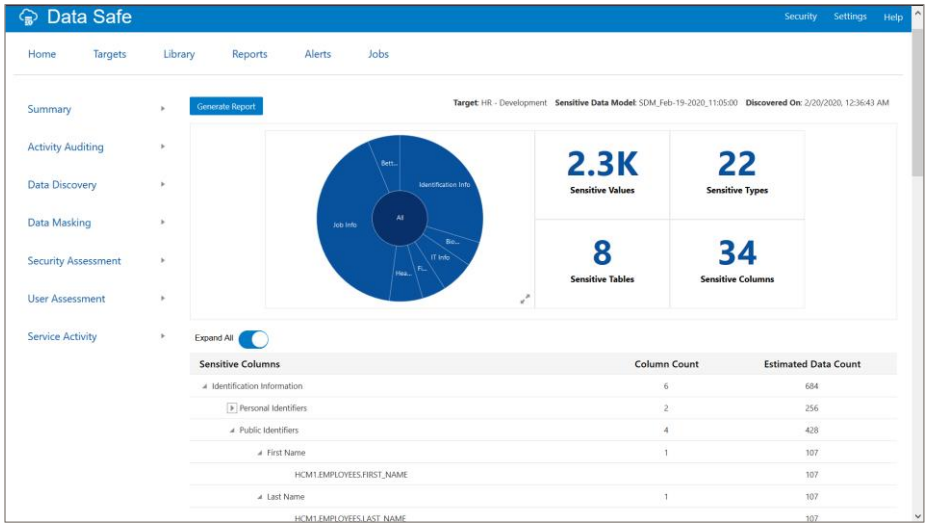


Figure 6: Sensitive data discovery feature report in Data Safe

Data Safe is just one way of scanning for sensitive data. One can optionally use DBSAT which outputs the report of sensitive data summary as shown in the sample below:

Schema	Table Name	Columns	Sensitive Columns	Rows	Sensitive Category
HCM_USER	EMPLOYEES	11	8	107	IDENTIFICATION INFO – PUBLIC IDS, JOB INFO – COMPENSATION DATA, JOB INFO – EMPLOYEE DATA, JOB INFO – ORG DATA

Figure 7: DBSAT report of database tables containing sensitive data

Oracle Enterprise Manager's component–Application Data Modeling (ADM) can also help discover columns containing sensitive information and identify the referential or parent-child relationships between these columns.

Once you understand your sensitive data landscape, you should determine who can access your sensitive data.

Identify who are the authorized database users accessing the sensitive data. Attempts by anyone else to access sensitive data will need to be vigilantly watched. Note that users with administrative privileges are privileged users and should be audited differently than non-administrative database users. The earlier section on Privileged user activity auditing covered the best practices for auditing administrative database users.

Authorized user access to sensitive data represents a list of database users with a valid business reason to access data, typically referred to as an allow-list. These users may be application service accounts, used by an application to perform a defined set of standardized business functions; or database users granted access to the database to generate reports, perform ad-hoc queries or otherwise interact with data.

For **application service accounts**, you may want to focus on trusted path when designing your audit policy. By trusted path, we mean the collection of session attributes that help define a trusted access path to the data. For example, the trusted path for an activity could be access that originates from a known set of IP addresses, using a pre-approved program, using a well-defined set of application context, or running as a designated operating system user. Since one of the common ways that databases are breached is through compromised login credentials, knowing the trusted path for your application makes it easier to focus your auditing on inappropriate use of application credentials. For many organizations, data access through the trusted path presents a lower level of risk, and therefore

needs a lower level of auditing, or may not need to be audited at all. Data access by an application service account outside of the trusted path indicates the account is being used for something other than application access, and therefore presents a higher level of risk with an accompanying higher level of audit requirement. Unless your organization explicitly allows the use of application service accounts for ad-hoc access to data by human actors, you should fully audit all activity by the application accounts outside of the trusted path.

For **database users** authorized to interact directly with data, auditing is strongly recommended as there is no intermediate application layer mediating access to the raw database tables.

Attempts by **anyone else** to access sensitive data presents the highest level of risk and should always be audited. The most effective way to define such users is to simply create an allow-list of database users and then audit any activity generated outside of that list. In rare cases, you may want to explicitly target individual accounts for higher levels of auditing. Create a watch-list of such individual database accounts who need to be vigilantly monitored for access. This type of auditing is typically required while investigating activities by suspicious employees or contractors. Such granular auditing is also often appropriate if you allow a third-party organization to directly interact with your database.

Now that you understand your sensitive data landscape and the users who are authorized to access them, you can configure audit policies for sensitive data access.

AUDIT FOR SENSITIVE DATA ACCESS
• Audit user access to sensitive data through untrusted path
• Audit user access to sensitive data
• Audit sensitive columns storing Personally Identifiable Information (PII) data

Table 13: Audit policies for sensitive data access

Let us examine in detail, the audit policies configuration for each of the above scenario.

Audit user access to sensitive data through untrusted path

For application service accounts, create an audit policy to audit action excluding the trusted paths or actions on non-sensitive objects. In this way, you are auditing application service account outside of the trusted path, which indicates that the account is being used for something other than application access.

A sample policy auditing all top-level actions on the selected tables in the HR schema outside of the trusted application path might look like the below one, where APPUSER_CONTEXT.APP_USER represents the application context values that are set in the database session by the application when Employees/ HRs/ HR Managers log in to the application respectively, and hence represents the trusted path. Any client connections which access these tables, but do not have the listed application context values set in their database session will be treated as untrusted and audited.

```
CREATE AUDIT POLICY USER_ACTIVITY_NOT_IN_TRUSTED_PATH
ACTIONS
ALL ON HR.EMPLOYEES
       , ALL ON HR.JOB_HISTORY
       , ALL ON HR.DEPARTMENTS
       , ALL ON HR.COUNTRIES
       , ALL ON HR.LOCATIONS
       , ALL ON HR.REGIONS
       , ALL ON HR.JOBS
WHEN 'SYS_CONTEXT("APPUSER_CONTEXT'', ''APP_USER'') NOT IN (''EMPLOYEE_USER'', ''HR_USER'', ''HR_MANAGER'')'
EVALUATE PER STATEMENT
ONLY TOPLEVEL;

AUDIT POLICY USER_ACTIVITY_NOT_IN_TRUSTED_PATH BY USERS WITH GRANTED ROLES  EMP_ROLE, HR_ROLE, HR_MGR;
```

For more secure trusted path validation checks, you can depend upon attributes that cannot be set by the application such as USERENV. Notice that the audit policy is enforced on application service accounts such as employees, HRs and HR Managers who are granted EMP_ROLE, HR_ROLE, and HR_MGR database roles respectively.

For users authorized to directly interact with sensitive data, minimally audit all top-level actions on sensitive data irrespective of trusted paths. A sample policy might look like:

```
CREATE AUDIT POLICY USER_ACTIVITY_HUMAN_ACTORS
ACTIONS
       ALL ON HR.EMPLOYEES
       , ALL ON HR.JOB_HISTORY
       , ALL ON HR.DEPARTMENTS
       , ALL ON HR.COUNTRIES
       , ALL ON HR.LOCATIONS
       , ALL ON HR.REGIONS
       , ALL ON HR.JOBS
       ONLY TOPLEVEL;

AUDIT POLICY USER_ACTIVITY_HUMAN_ACTORS by sophie, john;
```

Where the database users “sophie” and “john” have been granted direct access to the database to generate reports.

Note: The clause ‘ACTIONS ALL’ generates audit record for any operations on them including: ALTER, DELETE, GRANT, INSERT, SELECT, UPDATE, and CREATE. You can limit audit records to certain operations such as UPDATE.

Audit user access to sensitive data

Attempts by **anyone else** who is not authorized to access sensitive data, or in the watch-list should always be audited. Track all actions of such users on sensitive objects. A sample policy of auditing modification of sensitive data by users outside of the allow-list or the application service account is shown below. Here, the database user “hr_ann” represents the HR Manager authorized to do DML operations on DEPARTMENTS, COUNTRIES, LOCATIONS, REGIONS and JOBS in HR schema. But modification attempts by anyone else will be tracked, even if they are coming through valid application connection paths.

```
CREATE AUDIT POLICY REJECT_LIST_ACTIVITY
ACTIONS
       INSERT ON HR.DEPARTMENTS, UPDATE ON HR.DEPARTMENTS, DELETE ON HR.DEPARTMENTS
       , INSERT ON HR.COUNTRIES, UPDATE ON HR.COUNTRIES, DELETE ON HR.COUNTRIES
       , INSERT ON HR.LOCATIONS, UPDATE ON HR.LOCATIONS, DELETE ON HR.LOCATIONS
       , INSERT ON HR.REGIONS, UPDATE ON HR.REGIONS, DELETE ON HR.REGIONS
       , INSERT ON HR.JOBS, UPDATE ON HR.JOBS, DELETE ON HR.JOBS
       WHEN 'SYS_CONTEXT(''APPUSER_CONTEXT'', ''APP_USER'') IN (''HR_USER'')'
       EVALUATE PER STATEMENT
       ONLY TOPLEVEL;

AUDIT POLICY REJECT_LIST_ACTIVITY EXCEPT hr_ann;
```

Audit sensitive columns storing Personally Identifiable Information (PII) data

If there is a requirement to do granular monitoring of sensitive data access such as the following scenarios, configure fine-grained auditing (FGA) policies to augment intrusion detection

1. Monitor access to security-relevant columns that hold sensitive PII information
2. Monitor the data access based on security-relevant column values
3. Customize audit settings such as accessing a table between 9 p.m. and 6 a.m. or on Saturday and Sunday
4. Alert the security administrator when an audited column that should not be changed at midnight is updated

A sample FGA policy to track anyone trying to update values in SALARY column of EMPLOYEES table in HR schema might look like:

```
BEGIN
DBMS_FGA.ADD_POLICY(
       object_schema      => 'HR',
       object_name        => 'EMPLOYEES',
       policy_name        => 'updates_on_salary_column',
       audit_column       => 'SALARY',
       enable             =>  TRUE,
       statement_types    => 'UPDATE');
END;
```

You can optionally fine-tune FGA audit policy using an *audit_condition* parameter such that one can audit data access to specific rows based on threshold. You can also configure event handlers with *handler_schema* and *handler_module* parameters that could trigger alerts to administrators on policy violation.

Refer to the chapter 'Auditing Specific Activities with Fine-Grained Auditing' for configuring FGA audit policies in the Oracle Database Security Guide.

If there is a requirement to track value changes of such security-relevant columns for forensics/compliance, AVDF provides Data Modification Before-After Values Report as shown below to monitor such sensitive data updates.

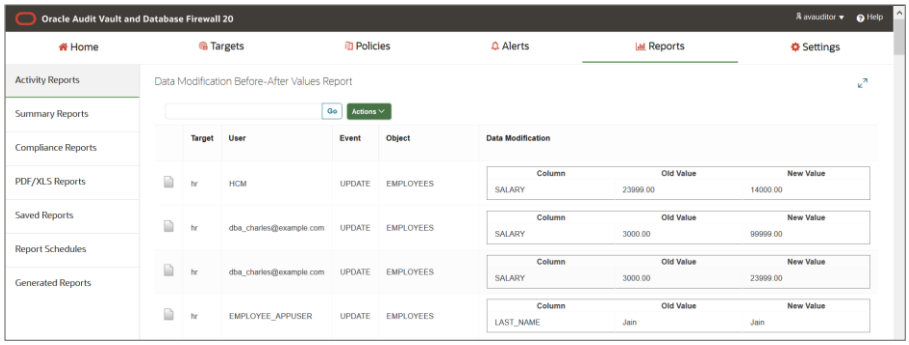


Figure 8: Data Modification Before-After Values Report in AVDF

Summary of audit policies and provisioning

In the prior sections, we recommended audit policies based on the most common security relevant actions in the database. These recommendations can be implemented using **Oracle Audit Vault and Database Firewall (AVDF), Oracle Data Safe** or **Oracle Database**. Some of the audit configurations would be unique to your scenario (e.g. sensitive data access), and we recommend you create custom audit policies in **Oracle Database**. Refer to the table below for a summary of the recommended policies and their provisioning options.

AUDIT TYPE	AUDIT POLICIES	AVAILABLE OUT-OF-THE-BOX	SUPPORTED BY AVDF/DATA SAFE?	SUPPORTED VIA CUSTOM UNIFIED FGA POLICIES
Privileged user activity auditing	Audit administrative database user accounts	No	Yes Admin Activity Auditing policy	Yes [1]
	Audit database user accounts with direct database access	No	No	Yes
	Audit high risk database user accounts	No	Yes User Activity Auditing policy	Yes [1]
Security-relevant events auditing	Audit security-management events	Yes ORA_SECURECONFIG and ORA_ACCOUNT_MGMT	Yes [3] Critical Database Activity policy	Yes [1]
	Audit account-management events	Yes ORA_SECURECONFIG and ORA_ACCOUNT_MGMT	Yes [3] Critical Database Activity policy	Yes [1]
	Audit data-security events	Yes Mandatory auditing, ORA_RAS_POLICY_MGMT, ORA_RAS_SESSION_MGMT, ORA_SECURECONFIG	Yes[3] Predefined Oracle Database policies	Yes
	Audit database-management events	Yes Mandatory auditing, ORA_SECURECONFIG	Yes [3] Admin Activity Auditing policy	Yes
	Audit data-management events	No	Yes	Yes [1]

			Database Schema Changes policy	
	Audit activities with system privileges	No	No	Yes
	Audit activities of unused system privileges	No	No	Yes
	Audit usage of components with data implications	Yes ORA_RAS_POLICY_MGMT, ORA_DV_AUDPOL, ORA_DV_AUDPOL2	Yes [3] Predefined Oracle Database policies	Yes [2]
	Monitor suspicious user-activity events: Multiple failed login attempts	Yes ORA_LOGON_FAILURES	Yes [3] Login Events policy	Yes [1]
	Monitor suspicious user-activity events: Sudden activity in dormant accounts	No	No	Yes
	Monitor suspicious user-activity events: Non-business hour activities	No	No	Yes
Sensitive data access auditing	Audit user access to sensitive data through untrusted path	No	No	Yes
	Audit user access to sensitive data	No	No	Yes
	Audit sensitive columns storing personally identifiable information (PII) data	No	No	Yes

Table 14: Summary of recommended audit policies and their provisioning options

Notes:

Yes [1]: Since the policy is supported by Data Safe/AVDF, consider provisioning from either of them. Consider configuring custom policies in Oracle database only if you are not using AVDF/Data Safe, or you want to configure and manage them selectively based on your specific needs.

Yes [2]: Provision custom unified audit policy in Oracle Database if you want to audit usage of any component that is not already covered in the predefined Oracle Database policy.

Yes [3]: AVDF/Data Safe can provision predefined Oracle Database policies. If there is a recommended policy in AVDF/Data Safe, consider provisioning them unless it is already provisioned in your target.

Auditing for compliance needs

As organizations across the globe need to comply with security frameworks like STIG and CIS to reduce risk levels and prevent or mitigate cyberattacks, there are compliance-specific audit requirements. Leverage the following predefined audit policies of Oracle Database mentioned in the appendix section Predefined unified audit policies of Oracle Database, to help accelerate your compliance:

- Security Technical Implementation Guides (STIG) compliance
- Center for Internet Security (CIS) compliance

Note: Compliance audit policy can also be provisioned from Data Safe and AVDF console.

Provisioning options

Oracle Audit Vault and Database Firewall (AVDF) and Data Safe can provision predefined audit policies discussed in prior sections. These policies are broadly classified into the following categories

1. Basic auditing policies
    a. Critical Database Activity
    b. Login Events
    c. Database Schema Changes
2. Administrator and user activity auditing policies
    a. Admin Activity Auditing
    b. User Activity Auditing
3. Audit Compliance Standards
4. Oracle Predefined Policies
    a. Predefined Oracle Database policies listed in the appendix section Predefined unified audit policies of Oracle Database
    b. Autonomous Databases may have additional predefined policies
5. Custom Policies
    a. Retrieve custom unified audit policies from the Oracle Database, and enable/disable them from the console

The unified audit policy provisioning in AVDF console is accessible from Auditor's page:

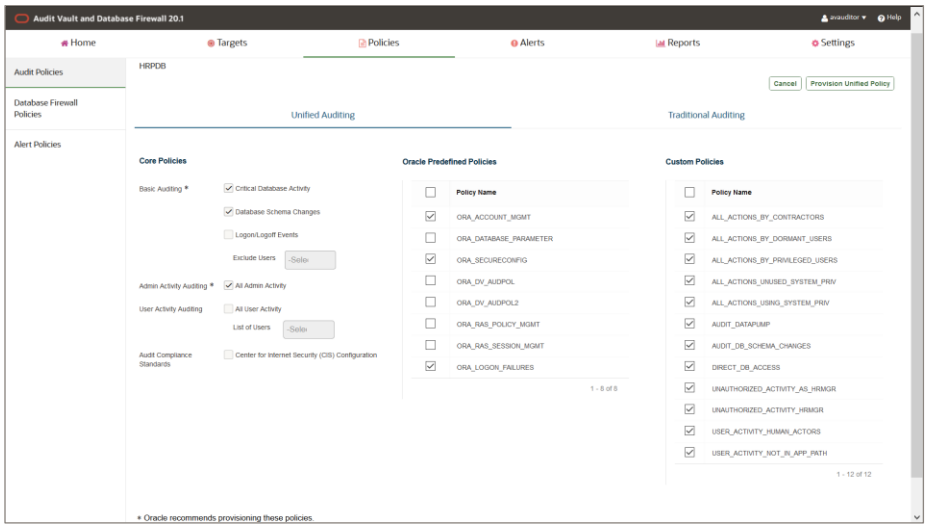


Figure 9: Unified audit policy provisioning in AVDF20

Refer to the AVDF Auditor's documentation guide to learn more details about provisioning the audit policies.

Audit policies can be easily provisioned from Data Safe as shown:

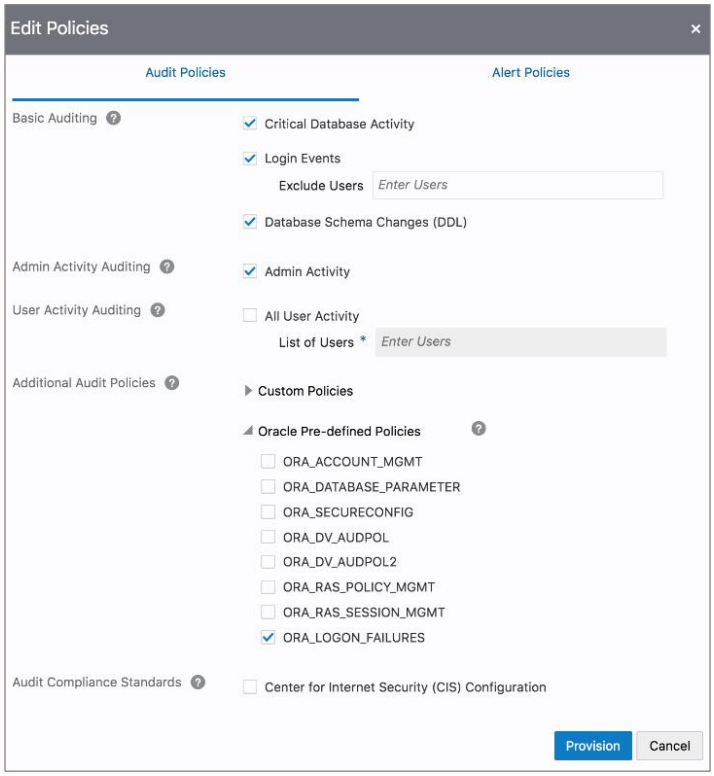


Figure 10: Unified audit policy provisioning in Data Safe

Audit trail management

It is important to properly manage audit trail on your databases to ensure efficient performance and optimum use of the disk space. As audit trails on your databases grow in volume, querying the audit trail with large volume of audit data may impact performance and lead to space scalability issues. It is best to archive the old records and then purge them from the online audit trail periodically.

Because database audit trails are typically stored in the SYSAUX tablespace, they can potentially fill it up and could start affecting other database operations that rely on the SYSAUX tablespace. In the eventuality of SYSAUX tablespace becoming full, audit record write will spillover to OS audit files and post a message to ALERT LOG so that corrective action can be initiated. If the OS file system space also becomes full, all database operations will start to fail. In addition, there is manageability overheard involved in loading back the spillover files to database tables.

To address these challenges, we make the following recommendations:

1. Relocate the unified audit trail table to a dedicated tablespace
2. Set a reasonable unified audit trail partition interval
3. Archive audit records and purge the unified audit trail
4. Improving query performance on the unified audit trail

Let us examine these recommendations in detail.

Relocate the unified audit trail table to a dedicated tablespace

Storing audit records in a separate dedicated tablespace will improve system performance and eliminate the probability of some other component taking up the free space. You can designate a different tablespace on a running instance, including one that is encrypted, by using the DBMS_AUDIT_MGMT.SET_AUDIT _TRAIL_LOCATION procedure. This procedure sets the tablespace for newer audit records but does not move the older audit records.

The new tablespace must be created as an ASSM (auto segment space managed) tablespace.

Set reasonable unified audit trail partition interval

Set the audit trail partition interval such that each partition has manageable set of audit records. This helps in both reading audit records (with partition pruning, when queried using EVENT_TIMESTAMP_UTC column of UNIFIED_AUDIT_TRAIL view), as well as audit trail cleanups since older partitions can be dropped much more quickly than deleting partial set of rows in a partition.

AUDSYS.AUD$UNIFIED table is interval partitioned with default interval of
1 month until 19c, and default interval of 1 day above 19c. If you have a high audit record rate and are using 1 month partition interval, then too many audit records may be generated in the same partition, which can negatively affect query performance against that partition. In this case, you should change the interval frequency to a shorter interval, such as one week or one day.

For instance, the sample below sets the partition interval to occur every 1 week:

```
BEGIN
DBMS_AUDIT_MGMT.ALTER_PARTITION_INTERVAL(
       interval_number       => 7,
       interval_frequency    => 'DAY');
 END;
```

The next partition is created only after current/active partition's HIGH_VALUE is reached in AUDSYS.aud$unified table. Therefore, it might take a while for the newer partition to appear.

Archive audit records and purge the unified audit trail

To maintain the integrity and reliability of audit data, keep only minimal required audit data locally and move audit data to a dedicated repository outside of the source database (such as AVDF or Data Safe) for long-term audit data retention and detailed analysis.

Periodically, archive and then purge old audit records at source especially when the number of records in the unified audit trail begins to reach a significantly large number. You can purge a subset of audit trail records or create a purge job that performs cleanup at a specified time interval. The DBMS_AUDIT_MGMT package provides utilities to set archive timestamp, purge the audit trail and schedule a purge job.

For instance, the sample below calls the CREATE_PURGE_JOB procedure to create a cleanup job for all audit trail types. The cleanup job is invoked every 100 hours, and all audit records older than the last archive timestamp are deleted.

```
BEGIN
DBMS_AUDIT_MGMT.CREATE_PURGE_JOB(
       audit_trail_type             =>  DBMS_AUDIT_MGMT.AUDIT_TRAIL_ALL,
       audit_trail_purge_interval   =>  100 /* hours */,
       audit_trail_purge_name       =>  'CLEANUP',
       use_last_arch_timestamp      =>  TRUE);
END;
```

*Note: **AVDF and Data Safe** are integrated with the DBMS_AUDIT_MGMT package on Oracle Databases. This integration manages the purging of audit records from unified audit trail after they are successfully inserted into their repository. Auto-purge feature in Data Safe when enabled, automates purging of audit records from unified audit trail.*

Improving query performance on the unified audit trail

In-order to query unified audit trail quickly, adhere to the following guidelines:

1. If unified audit records have been written to operating system spillover files, load them to the unified audit trail.

   When the database is not writable (such as standby databases operating in mount mode), or if the database is closed, or read-only, then Oracle Database writes audit records to external files called spillover audit files. This enables auditing of actions by privileged users during operations like backup, recovery, etc. These external files are created in the $ORACLE_BASE/audit/$ORACLE_SID directory. You can load these files into the database by running the DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES procedure. You can do this either explicitly or by configuring an Oracle Scheduler job. Ensure the audit records are loaded into the database table for performant queries on UNIFIED_AUDIT_TRAIL view.

2. Periodically gather statistics on the unified audit internal table to ensure the statistics are up to date.

   Execute the DBMS_STATS.GATHER_TABLE_STATS procedure on the AUD$UNIFIED table in the AUDSYS schema periodically to ensure that the unified audit table statistics are updated. The frequency of periodic execution is a factor of audit event generation rate in the database. The higher the audit event generation rate, the more frequently you should collect statistics.

3. When you query the UNIFIED_AUDIT_TRAIL data dictionary view, include the EVENT_TIMESTAMP_UTC column in a WHERE clause.

   The EVENT_TIMESTAMP_UTC column records the timestamp for audited events in UTC time zone. This column is the partition key for AUD$UNIFIED table. Including this column when querying the audit trail helps to achieve partition pruning, and thus improves read performance of the UNIFIED_AUDIT_TRAIL view.

Summary

Database auditing is an integral component of an organization's data security architecture, enabling organizations to monitor and detect suspicious database activities in addition to addressing compliance requirements. Effective auditing requires that audit policies be selective and focused to ensure that the audit records generated are what is needed to support forensic analysis and/or compliance, without generating unnecessary audit records.

Oracle Database provides the industry's most comprehensive auditing capabilities, collecting detailed information on critical events. With effective auditing principles focused on the privileged user activity, security-relevant events, and sensitive data access, administrators can provision precise, context-aware unified audit policies. Oracle Database, Oracle Audit Vault and Database Firewall (AVDF 20) and Data Safe provide several predefined unified audit policies, simplifying the provisioning and management process. In addition to audit policies, audit trail management is an important aspect to be considered and best practices pertaining to audit trail management should be leveraged.

Appendix

Glossary of terms

TERM	DESCRIPTION
Database Auditing	Auditing is the monitoring and recording of configured database actions.
Auditable event	A single action that triggers generation of audit data.
Audit record	Security-relevant record that provides documentary evidence of an auditable event. Audit record includes who did what, when, where and how.
Audit trail	The location where audit records are written. This might be a database table, an external system like SYSLOG, the Windows Event Log, or a file in the operating system.
Audit policy	A set of SQL statements that describe which events will generate an audit record.
Audit facility	One of several broad collections of audit features within the database - examples include unified audit, traditional audit, fine-grained audit, and Database Vault audit.
Audit Vault and Database Firewall (AVDF)	Software appliance that provides a complete Database Auditing and Activity Monitoring solution that combines native audit logs with network traffic capture for Oracle and non-Oracle databases.
Data Safe	A cloud service that provides an integrated, unified set of features that include security assessment, user risk assessment, sensitive data discovery, sensitive data masking, audit policy control, audit record collection, reporting, and alerting.
Database Security Assessment Tool (DBSAT)	Stand-alone command line tool that scans the database to evaluate the database's security posture. DBSAT also scans for sensitive and personal data using customizable regular expression patterns, and reports on the amount and type of sensitive data found.

Table 15: Terms and definitions

Benefits of unified audit over traditional auditing

Significant benefits of unified audit over traditional auditing are summarized in the table below:

KEY AREA	FEATURE	TRADITIONAL AUDIT	UNIFIED AUDIT
Configurability	Creation and Management of what to audit	Each audit instruction (statement, privilege, and object) managed and configured individually.	With unified audit, instructions on how to record auditable events are grouped together into named audit policies, which can be enabled, disabled, and redefined. Named audit policies can be created once and enforced in multiple dimensions (e.g. on users, roles), giving a lot more flexibility and simplicity.
	Selective and conditional auditing to focus on relevant activity	Not available	With conditional auditing, you can create precise, highly selective and context-aware policies, which makes it easier to audit specific actions and reduce the amount of irrelevant audit records. This lowers storage needs and provides high-value audit records that will be useful for auditors, forensic investigations or regulatory compliance needs. Conditions can be based on Application Contexts, session context, and built-in functions.

	Consume pre-defined audit policies	There are very few security-relevant SQL statements and privileges that are audited by default.	There are several pre-created "best practice" audit policies that reduce effort to design effective auditing. For those organizations that follow a widely used security framework (for example, the CIS Benchmark), we provide out-of-box audit policies that make it easy to conform to that framework.
Consolidation	Audit trail consisting of all the audit data	Different database components write to multiple audit trails, and create audit records with different formats using different data types. Obtaining a comprehensive view of database activity across all the audit sources and trails was a complex task, as was managing the audit configuration.	Unified audit combines all audit trails into a single unified audit trail. Audit records are generated by a variety of audit sources including: a. Audit system related sources: Audit records (including SYS audit records), Mandatory audit records b. Security-control related sources: Oracle Database Vault, Oracle Label Security, Oracle Real Application Security c. Database operations related sources: Oracle Recovery Manager, Oracle Data Pump, Oracle SQL*Loader With unified audit, audit records from all audit sources are written to a consolidated audit trail – AUDSYS.AUD$UNIFIED table or OS files, and exposed through the UNIFIED_AUDIT_TRAIL view. The unified audit trail also normalizes the audit record format, using standardized column names and data types across all audit sources. The consolidated, normalized unified audit trail simplifies collection, analysis, and management of audit records generated by the different audit sources. Consistent formatting simplifies reporting and analysis of the audit data.
Extensibility	Extensible to capture application attributes	Not available	Unified audit trail can be extended to include application attributes by configuring auditing for application context values. Application context namespace can be populated with the required attributes, and this is captured in the APPLICATION_CONTEXTS column of the unified audit trail.
Security	Integrity of audit trail ensuring that users cannot tamper with the audit trail	All DELETE, INSERT, UPDATE, and MERGE on the audit trail SYS.AUD$ table are always audited, and such audit records are not allowed to be deleted by non-SYS user. Only the user SYS, or a user to whom SYS has granted DELETE privilege on SYS.AUD$ can delete records from the database audit trail SYS.AUD$.	Unified audit trail is stored in a storage only schema for AUDSYS, where no one is allowed to login to that schema in the database. AUD$UNIFIED is a specialized table which allows only INSERT activity. Any attempt to directly truncate, delete or update contents of AUD$UNIFIED fails, and any such attempt generates an audit record. Audit data is managed using the built-in audit data management DBMS_AUDIT_MGMT package. Additionally, audit tablespace can be encrypted with Transparent Data Encryption (TDE). The unified audit table can also be protected with a Database Vault realm. We recommend you create a Database Vault realm around AUDSYS schema

			so that only authorized administrators can access the unified audit trail.
Integrity	Ability to introduce audit integrity checks	There is a single source of truth for audit data. Audit data can either be written to database audit trail, or into syslog location by setting syslog parameter.	The integrity checks of audit data is more complete in unified audit since there are two reliable sources of truth for audit data. With the UNIFIED_AUDIT_SYSTEMLOG parameter set, the audit records are not only written to the database audit trail, but in-addition, certain key fields are written to syslog. As syslog is immutable by SYSDBA, audit data integrity is more reliable.

Table 16: Benefits of unified audit over traditional audit

Auditing functionality by releases

Oracle Database's audit features grew over almost three decades of development. A few milestones are highlighted from the development of the Oracle Database audit feature:

ORACLE DATABASE RELEASE (YEAR OF RELEASE)	AUDITING FEATURES INTRODUCED/ENHANCED
Oracle 7 (1992)	In Oracle 7, Oracle Database auditing was introduced with support for three different audit facilities - Default mandatory auditing, traditional auditing (for Operating system and Oracle Database) and value-based auditing with triggers. 1. Default (Mandatory) auditing ensured some database events are always audited and cannot be disabled. These events include Database startup, Database shutdown, and Administrative Privileged (SYSDBA, SYSOPER, etc.) connections. Audit records are written to either of two audit trails: 2. Operating System Event Log (Windows) 3. AUDIT_FILE_DEST for most other platforms and the value defaults to $ORACLE_BASE/admin/$ORACLE_SID/adump. For startup records before initialization parameter file is read, the value defaults to $ORACLE_HOME/rdbms/audit. 4. Traditional auditing captures audit information at statement, privilege or object level. Traditional auditing can be conditional within narrow boundaries. 5. With statement auditing, administrators can audit selected DDL and DML commands like create, alter, delete, truncate, drop, select etc. 6. With privilege auditing, administrators can audit usage of any system privilege like select any table, create any trigger, alter any procedure etc. 7. With object auditing, administrators or the object's owner can audit DML commands, including queries, against an object and execution on owned PL/SQL procedures or functions. 8. Audit records generated by the traditional auditing facility are stored in either of two audit trails, which is determined by the parameter AUDIT_TRAIL: • Database (sys.aud$) table for all Operating Systems • OS Event Log (Windows) 9. Value-based auditing with triggers for auditing table object provided crucial information on what records were changed or what values were changed. The audit record format (including the contents of the record) and the audit trail used, typically a database table, is determined by the trigger code. Reference for Oracle 7 database auditing is here: https://www.oracle.com/servers/technologies/oracle7.html
Oracle 9iR2 (2002)	Several enhancements were introduced, including 1. A new method of value-based auditing was added, leveraging redo transaction logs with LogMiner and Oracle Streams. 2. Default (Mandatory) auditing was enhanced to capture all top-level SQL statements issued using SYSDBA or SYSOPER administrative privileges if the initialization parameter AUDIT_SYS_OPERATIONS is set to TRUE. By default, AUDIT_SYS_OPERATIONS is set to FALSE (this later switched to TRUE in the 12.1 database release).

	3. Fine-grained auditing (FGA) was introduced, expanding the capability to audit SELECT on columns that a SQL statement has accessed, and could be conditional based upon values of data being selected. Event handler integration in FGA provides flexibility in alerting the audited event. Reference for Oracle 9i enhancements in auditing is here: https://docs.oracle.com/cd/B10501_01/server.920/a96531/ch3_9ir1.htm - 78308
Oracle 10gR1 (2003)	The fine-grained auditing (FGA) facility was enhanced to support granular auditing of queries as well as UPDATE, INSERT, and DELETE operations. Transactions and SQL information also were added to the audit tables to further improve accountability of all users. Reference for Oracle 10gR1 enhancements in auditing is here: https://docs.oracle.com/cd/B14117_01/server.101/b10750/chapter1.htm#sthref470
Oracle 10gR2 (2005)	Added a new XML-based audit trail with location of XML files controlled by AUDIT_FILE_DEST. Optional extended attributes enhanced the audit records in the traditional auditing facility. Extended attributes included SQL text and bind variable values. A common audit trail DBA_COMMON_AUDIT_TRAIL was introduced to present both the traditional and the fine-grained audit log records in a single view. It was the first attempt to unify the audit trails into a common view. Reference for Oracle 10gR2 enhancements in auditing is here: https://docs.oracle.com/cd/B19306_01/server.102/b14214/chapter1.htm#AREANO02020
Oracle 11gR2 (2009)	Several enhancements in DBMS_AUDIT_MGMT package to better manage the audit trail cleanup and audit data management, including controlling the size and age of the audit trail written to operating system files, and moving the database audit trail tables outside of the default SYSTEM tablespace. Reference for Oracle 11gR2 enhancements in auditing is here: https://docs.oracle.com/cd/E11882_01/network.112/e36292/whatsnew.htm#DBSEG000
Oracle 12cR1 (2013)	Unified audit was introduced with the following primary objectives: 1. Consolidate all of the existing audit trails (traditional, default, FGA, Database Vault, Label Security, etc.) into a single audit trail. 2. Introduce a new schema, AUDSYS, for the unified audit trail data table. 3. Make the audit trail more secure and tamper resistant. 4. Streamline audit management. 5. Streamline audit administration with Separation of Duties. 6. Reduce audit implementation effort with pre-configured “best practices” audit policies 7. Improve audit performance 8. Make audit more selective Reference for Oracle 12cR1 enhancements in auditing is here: https://docs.oracle.com/database/121/DBSEG/release_changes.htm#GUID-1506B929-1903-4CD5-BFC9-3911B5863A16
Oracle 12cR2 (2017)	Enhancements include 1. Define unified audit policies that conditionally audit users based on a role. 2. Enhance security measures and better-read performance for the AUDSYS schema, which stores unified audit records. 3. Integrate Transparent Sensitive Data Protection (TSDP) with unified audit and fine-grained audit. Reference for Oracle 12cR2 enhancements in auditing: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/dbseg/release-changes.html#GUID-F9A84B43-9283-495E-A6E0-5F55B1E7A531
Oracle 18c (2018)	Enhancements include 1. Include Unified audit as part of a full database export or import operation using Oracle Data Pump. 2. Write unified audit records to SYSLOG on UNIX or the Windows Event Viewer on Microsoft Windows by setting UNIFIED_AUDIT_SYSTEMLOG parameter.

	Reference for Oracle 18c enhancements in auditing: https://docs.oracle.com/en/database/oracle/oracle-database/18/dbseg/release-changes.html#GUID-B41ACBF0-79B8-4D85-AD04-14D551848637
Oracle 19c (2019)	Enhancements include Allow auditing only the top-level SQL commands issued by the user, but not the resulting recursive SQL. For example, if an administrator issues a command to collect statistics for a database schema with top-level auditing, audit record would be generated for that command, but not for the automatically generated SELECT statements for each object within the schema. Top-level statement auditing significantly reduces the volume of audit records generated for certain activities. 1. Partitioned the unified audit default audit trail, and added a column EVENT_TIMESTAMP_UTC to eliminate partitions while reading from the audit trail, thus improving read performance. 2. Include the pluggable database identifier in audit records Reference for Oracle 19c enhancements in auditing: https://docs.oracle.com/en/database/oracle/oracle-database/19/dbseg/release-changes.html#GUID-B088E9B3-31EE-46A7-95D3-A7F1088EC5CF

Table 17: Auditing functionality enhancements by releases

Since 2013, most new auditing enhancements have been focused on the unified audit facility, and our path forward, and in upcoming releases of Oracle Database, we plan to de-support the traditional auditing facility.

Mandatory audit configurations of Oracle Database

Certain security sensitive database activities are always audited and they cannot be disabled. Do not create audit policies that duplicate the audit information already captured with mandatory audit.

The mandatory audit configuration includes:

- Top-level statements executed by users with administrative privileges such as SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSDG, and SYSKM, until the database opens
- Attempts to modify or delete audit records
- Attempts to modify the AUD$UNIFIED table in the AUDSYS schema
- Oracle Database Vault configuration changes
- Audit-related activities, such as modifications to audit policies and executions of DBMS_AUDIT_MGMT package

For a complete list of mandatory auditable events corresponding to your database version, refer to the section 'Activities That Are Mandatorily Audited' in the Oracle Database Security Guide.

Predefined unified audit policies of Oracle Database

Oracle Database provides several pre-designed "best practice" unified audit policies that cover common security-relevant audit settings as mentioned below. These vary by the database version, and therefore you should check the section 'Auditing Activities with the Predefined Unified Audit Policies' in the corresponding Oracle Database Security Guide.

POLICY NAME	PURPOSE
ORA_LOGON_FAILURES *	Audits failed logons only
ORA_SECURECONFIG *	Audits high-value, low-frequency DDL and DCL commands

ORA_DATABASE_PARAMETER	Audits changes to Oracle Database parameter settings
ORA_ACCOUNT_MGMT	Audits modifications to user accounts and privileges
ORA_CIS_RECOMMENDATIONS	Audit requirements for Center for Internet Security (CIS) compliance
ORA_RAS_POLICY_MGMT, ORA_RAS_SESSION_MGMT	Audits Real Application Security events
ORA_DV_AUDPOL	Audits Oracle Database Vault DVSYS, LBACSYS and DVF schema objects
ORA_DV_AUDPOL2	Audits Oracle Database Vault default realms and command rules
ORA_STIG_RECOMMENDATIONS **ORA_ALL_TOPLEVEL_ACTIONS** **ORA_LOGON_LOGOFF**	Audit requirements for Security Technical Implementation Guides (STIG) compliance

Table 18: Predefined unified audit policies of Oracle Database

* Enabled by default only for DBCA created databases.

We recommend that you enable ORA_LOGON_FAILURES and ORA_SECURECONFIG in your non-DBCA created databases. Enable other predefined audit policies after evaluating your requirements.

For the detailed list of latest predefined unified audit policies and their policy configuration setting, refer to the section 'Auditing Activities with the Predefined unified audit Policies' in the Oracle Database Security Guide.

Configuration of sample audit policies

To configure the HR demo schema and the various audit policies discussed in the technical report, follow these steps:

1. Ensure you have Oracle Database 12c and above (either non-CDB or CDB with a PDB). The HR demo schema is shipped with Oracle Database installation.
2. Execute the Oracle script $ORACLE_HOME/demo/schema/human_resource/hr_main.sql. For instance, the below execution creates the HR schema in PDB instance named hrpdb in the tablespace USERS.

```
sqlplus system@ hrpdb @$ORACLE_HOME/demo/schema/human_resources/hr_main.sql password users temp /tmp
Enter password: password
```

3. Execute the script hrpdb_data_script.sql from the archive, connecting to the instance hrpdb as privileged user SYS. The script creates sample users, roles, application context and various audit policies.

   Note: If you have used a different pdb name than hrpdb, open the script and change PDB_NAME variable value. Provide the password of privileged user SYS during execution.

4. Execute the script hrpdb_workload.sql from the archive, connecting to the instance hrpdb as privileged user SYS. Provide SYS password that was used in step3, and HR user/password that was used in step2. The script runs various workload queries as different users, and queries, which are executed within trusted application paths and those, which are not.

   Note: To see the audit trail triggered by the workload in your sample schema, start with clean audit trail by executing DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL package as shown below.

```
BEGIN
DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL(
    audit_trail_type         =>  DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
    use_last_arch_timestamp  =>  FALSE);
END;
```

5. Query UNIFIED_AUDIT_TRAIL view for the results. You might have the query, which looks like below to see what unified audit policy was triggered for the SQL and for which database user.

```
select sql_text, unified_audit_policies, dbusername from unified_audit_trail order by event_timestamp_utc desc
```

Optionally, if AVDF or Data Safe is monitoring the pdb instance, you can navigate to All Activity report to see the audit events generated for different policies. The samples are for illustrative purposes only.

Author and acknowledgements

Author: Angeline Janet Dhanarani, Database Security Product Management

Acknowledgements: Gratefully acknowledge Vipin Samar, Senior Vice President Database Security for his immense support, guidance and encouragement. Sincerely appreciate valuable feedbacks from Russ Lowenthal, Gopal Mulagund, Rajesh Tammana, Abhishek Munnolimath and Ashok Swaminathan.

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle