A

Keeping Your Oracle Database Secure

Oracle provides guidelines for keeping your database secure, such as advice on securing user accounts, privileges, roles, passwords, and data.

- About the Oracle Database Security Guidelines
  Information security, and privacy and protection of corporate assets and data are critical in any business.
- Downloading Security Patches and Contacting Oracle Regarding Vulnerabilities
  You should always apply security patches as soon as they are available. If problems arise, then you should contact Oracle regarding vulnerabilities.
- Guidelines for Securing User Accounts and Privileges
  Oracle provides guidelines to secure user accounts and privileges.
- Guidelines for Securing Passwords
  Oracle provides guidelines for securing passwords in a variety of situations.
- Securing Authentication for Oracle Database Microsoft Windows Installations
  By default, the `SQLNET.NO_NTLM` parameter setting in the `sqlnet.ora` file on Microsoft Windows installations with `AUTHENTICATION_SERVICES=NTS` is `TRUE`.
- Guidelines for Securing Roles
  Oracle provides guidelines for role management.
- Guidelines for Securing Data
  Oracle provides guidelines for securing data on your system.
- Guidelines for Securing the ORACLE_LOADER Access Driver
  Oracle provides guidelines to secure the `ORACLE_LOADER` access driver.
- Guidelines for Securing a Database Installation and Configuration
  Oracle provides guidelines to secure the database installation and configuration.
- Guideline for Securing Multitenant PDBs from the Root in a Linux Environment
  In Linux, you can securely compartmentalize PDBs to manage their resources in containers called nests.
- Guidelines for Securing the Network
  Security for network communications is improved by using client, listener, and network guidelines to ensure thorough protection.
- Guideline for Securing External Procedures
  The `ENFORCE_CREDENTIAL` environment variable controls how an `extproc` process authenticates user credentials and callout functions.
- Guidelines for Auditing
  Oracle provides guidelines for auditing.
- Addressing the CONNECT Role Change
  The `CONNECT` role, introduced with Oracle Database version 7, added new and robust support for database roles.

A.1 About the Oracle Database Security Guidelines

Information security, and privacy and protection of corporate assets and data are critical in any business.

Oracle Database comprehensively addresses the need for information security by providing cutting-edge security features such as deep data protection, auditing, scalable security, secure hosting, and data exchange.

Oracle Database leads the industry in security. To maximize the security features offered by Oracle Database in any business environment, it is imperative that the database itself be well protected.

Security guidelines provide advice about how to configure Oracle Database to be secure by adhering to and recommending industry-standard and advisable security practices for operational database deployments. Many of the guidelines described in this section address common regulatory requirements such as those described in the Sarbanes-Oxley Act. For more information about how Oracle Database addresses regulatory compliance, protection of personally identifiable information, and internal threats, visit:

http://www.oracle.com/technetwork/topics/security/whatsnew/index.html

A.2 Downloading Security Patches and Contacting Oracle Regarding Vulnerabilities

You should always apply security patches as soon as they are available. If problems arise, then you should contact Oracle regarding vulnerabilities.

- Downloading Security Patches and Workaround Solutions
  Security patches apply to the operating system on which Oracle Database resides, Oracle Database itself, and all installed Oracle Database options and components.
- Contacting Oracle Security Regarding Vulnerabilities in Oracle Database
  You can contact Oracle Security regarding vulnerabilities in Oracle Database.

A.2.1 Downloading Security Patches and Workaround Solutions

Security patches apply to the operating system on which Oracle Database resides, Oracle Database itself, and all installed Oracle Database options and components.

- To download security patches and workaround solutions:
  - For security patches, periodically check the security site on Oracle Technology Network for details about security alerts released by Oracle at http://www.oracle.com/technetwork/topics/security/alerts-086861.html.
  - Check the Oracle Worldwide Support Service site, My Oracle Support, for details about available and upcoming security-related patches at https://support.oracle.com.

A.2.2 Contacting Oracle Security Regarding Vulnerabilities in Oracle Database

You can contact Oracle Security regarding vulnerabilities in Oracle Database.

- Contact Oracle Security using either of the following methods:
  - If you are an Oracle customer or an Oracle partner, use My Oracle Support to submit a Service Request on any potential Oracle product security vulnerability.
  - Send an email to `secalert_us@oracle.com` with a complete description of the problem, including product version and platform, together with any scripts and examples. Oracle encourages those who want to contact Oracle Security to employ email encryption, using our encryption key.

A.3 Guidelines for Securing User Accounts and Privileges

Oracle provides guidelines to secure user accounts and privileges.

1. **Lock and expire default (predefined) user accounts.**

   Oracle Database installs with several default database user accounts. Upon successful installation of the database, the Database Configuration Assistant automatically locks and expires most default database user accounts.

   If you perform a manual (without using Database Configuration Assistant) installation of Oracle Database, then no default database users are locked upon successful installation of the database server. Or, if you have upgraded from a previous release of Oracle Database, you may have default accounts from earlier releases. Left open in their default states, these user accounts can be exploited, to gain unauthorized access to data or disrupt database operations.

   You should *lock* and *expire* all default database user accounts. Oracle Database provides SQL statements to perform these operations. For example:

   ```
   ALTER USER ANONYMOUS PASSWORD EXPIRE ACCOUNT LOCK;
   ```

   Installing additional products and components after the initial installation also results in creating more default database accounts. Database Configuration Assistant automatically locks and expires all additionally created database user accounts. Unlock only those accounts that need to be accessed on a regular basis and assign a strong, meaningful password to each of these unlocked accounts. Oracle provides SQL and password management to perform these operations.

   If any default database user account other than the ones left open is required for any reason, then a database administrator (DBA) must unlock and activate that account with a new, secure password.

   If a default database user account, other than the ones left open, is required for any reason, then a database administrator (DBA) can unlock and activate that account with a new, secure password.

   Securing Oracle Enterprise Manager Accounts

   If you install Oracle Enterprise Manager, the `SYSMAN` and `DBSNMP` accounts are open, unless you configure Oracle Enterprise Manager for central administration. In this case, the `SYSMAN` account (if present) will be locked.

   If you do not install Oracle Enterprise Manager, then only the `SYS` and `SYSTEM` accounts are open. Database Configuration Assistant locks and expires all other accounts (including `SYSMAN` and `DBSNMP`).

2. **Discourage users from using the NOLOGGING clause in SQL statements.**

   In some SQL statements, the user has the option of specifying the `NOLOGGING` clause, which indicates that the database operation is not logged in the online redo log file. Even though the user specifies the clause, a redo record is still written to the online redo log file.

However, there is no data associated with this record. Because of this, using NOLOGGING has the potential for malicious code to be entered can be accomplished without an audit trail.

3. **Practice the principle of least privilege.**

   Oracle recommends the following guidelines:

   a. **Grant necessary privileges only.**

      Do not provide database users or roles more privileges than are necessary. (If possible, grant privileges to roles, not users.) In other words, the *principle of least privilege* is that users be given only those privileges that are actually required to efficiently perform their jobs.

      To implement this principle, restrict the following as much as possible:

      - The number of SYSTEM and OBJECT privileges granted to database users.
      - The number of people who are allowed to make SYS-privileged connections to the database.
      - The number of users who are granted the ANY privileges, such as the DROP ANY TABLE privilege. For example, there is generally no need to grant CREATE ANY TABLE privileges to a non-DBA-privileged user.
      - The number of users who are allowed to perform actions that create, modify, or drop database objects, such as the TRUNCATE TABLE, DELETE TABLE, DROP TABLE statements, and so on.

   b. **Limit granting the CREATE ANY EDITION and DROP ANY EDITION privileges.**

      To maintain additional versions of objects, editions can increase resource and disk space consumption in the database. Only grant the CREATE ANY EDITION and DROP ANY EDITION privileges to trusted users who are responsible for performing upgrades.

   c. **Re-evaluate the SELECT object privilege and SELECT ANY TABLE system privileges that you have granted to users.**

      If you want to restrict users to only being able to query tables, views, materialized views, and synonyms, then grant users the READ object privilege, or for trusted users only, the READ ANY TABLE system privilege. If in addition to performing query operations, you want users to be able to lock tables in exclusive mode or perform SELECT ... FOR UPDATE statements, then grant the user the SELECT object privilege or, for trusted users only, the SELECT ANY TABLE system privilege.

   d. **Restrict the CREATE ANY JOB, BECOME USER, EXP_FULL_DATABASE, and IMP_FULL_DATABASE privileges. Also restrict grants of the CREATE DIRECTORY and CREATE ANY DIRECTORY privileges.**

      These are powerful security-related privileges. Only grant these privileges to users who need them.

   e. **Restrict the BECOME USER privilege to users of Oracle Data Pump, and the DBMS_WORKLOAD_CAPTURE and DBMS_WORKLOAD_REPLAY packages.**

      The BECOME USER privilege is used only for the following subsystems:

      - Oracle Data Pump Import utilities impdp and imp, to assume the identity of another user to perform operations that cannot be directly performed by a third party (for example, loading objects such as object privilege grants). In an Oracle Database Vault environment, Database Vault provides several levels of required authorization that affect grants of BECOME USER.

- `DBMS_WORKLOAD_CAPTURE` and `DBMS_WORKLOAD_REPLAY` PL/SQL packages, as a required privilege to be granted to users who must use these packages.

If you use the `AUTHID CURRENT_USER` clause when invoking one of these subsystems (for example, in static references in PL/SQL code), then ensure that the `CURRENT_USER` is granted the `BECOME USER` privilege, either by a direct grant or through a role.

f. **Restrict library-related privileges to trusted users only.**

The `CREATE LIBRARY`, `CREATE ANY LIBRARY`, `ALTER ANY LIBRARY`, and `EXECUTE ANY LIBRARY` privileges, and grants of `EXECUTE ON` *`library_name`* convey a great deal of power to users. If you plan to create PL/SQL interfaces to libraries, only grant the `EXECUTE` privilege to the PL/SQL interface. Do not grant `EXECUTE` on the underlying library. You must have the `EXECUTE` privilege on a library to create the PL/SQL interface to it. However, users have this privilege implicitly on libraries that they create in their own schemas. Explicit grants of `EXECUTE ON` *`library_name`* are rarely required. Only make an explicit grant of these privileges to trusted users, and never to the `PUBLIC` role.

g. **Restrict synonym-related privileges to trusted users only.**

The `CREATE PUBLIC SYNONYM` and `DROP PUBLIC SYNONYM` system privileges convey a great deal of power to these users. Do not grant these privileges to users, unless they are trusted.

h. **Do not allow non-administrative users access to objects owned by the SYS schema.**

Do not allow users to alter table rows or schema objects in the `SYS` schema, because doing so can compromise data integrity. Limit the use of statements such as `DROP TABLE`, `TRUNCATE TABLE`, `DELETE`, `INSERT`, or similar object-modification statements on `SYS` objects only to highly privileged administrative users.

i. **Restrict permissions on run-time facilities.**

Many Oracle Database products use run-time facilities, such as Oracle Java Virtual Machine (OJVM). Do not assign all permissions to a database run-time facility. Instead, grant specific permissions to the explicit document the root file paths for facilities that might run files and packages outside the database.

Here is an example of a vulnerable run-time call, which individual files are specified:

```
call dbms_java.grant_permission('wsmith', 'SYS:java.io.FilePermission','<<ALL
FILES>>','read');
```

Here is an example of a better (more secure) run-time call, which specifies a directory path instead:

```
call dbms_java.grant_permission('wsmith', 'SYS:java.io.FilePermission','<<actual
directory path>>','read');
```

4. **Revoke access to the following:**

- The `SYS.USER_HISTORY$` table from all users except `SYS` and `DBA` accounts
- The `RESOURCE` role from typical application accounts
- The `CONNECT` role from typical application accounts
- The `DBA` role from users who do not need this role

5. **Grant privileges only to roles.**

Granting privileges to roles and not individual users makes the management and tracking of privileges much easier.

6. **Limit the proxy account (for proxy authorization) privileges to CREATE SESSION only.**
7. **Use secure application roles to protect roles that are enabled by application code.**

   Secure application roles allow you to define a set of conditions, within a PL/SQL package, that determine whether or not a user can log on to an application. Users do not need to use a password with secure application roles.

   Another approach to protecting roles from being enabled or disabled in an application is the use of role passwords. This approach prevents a user from directly accessing the database in SQL (rather than the application) to enable the privileges associated with the role. However, Oracle recommends that you use secure application roles instead, to avoid having to manage another set of passwords.
8. **Create privilege captures to find excessively granted privileges.** Privilege analysis captures the privileges that users and applications use, and then presents these in a format for easy analysis. From there, you can revoke unnecessary privileges if you want.
9. **Monitor the granting of the following privileges only to users and roles who need these privileges.**

   By default, Oracle Database audits the following privileges:

   - `ALTER SYSTEM`
   - `AUDIT SYSTEM`
   - `CREATE EXTERNAL JOB`

   Oracle recommends that you also audit the following privileges:

   - `ALL PRIVILEGES` (which includes privileges such as `BECOME USER`, `CREATE LIBRARY`, and `CREATE PROCEDURE`)
   - `DBMS_BACKUP_RESTORE` package
   - `EXECUTE` to `DBMS_SYS_SQL`
   - `SELECT ANY TABLE`
   - `SELECT` on `PERFSTAT.STATS$SQLTEXT`
   - `SELECT` on `PERFSTAT.STATS$SQL_SUMMARY`
   - `SELECT` on `SYS.SOURCE$`
   - Privileges that have the `WITH ADMIN` clause
   - Privileges that have the `WITH GRANT` clause
   - Privileges that have the `CREATE` keyword
10. **Use the following data dictionary views to find information about user access to the database.**

    - `DBA_*`
    - `DBA_ROLES`
    - `DBA_SYS_PRIVS`
    - `DBA_ROLE_PRIVS`
    - `DBA_TAB_PRIVS`
    - `DBA_AUDIT_TRAIL` (if standard auditing is enabled)
    - `DBA_FGA_AUDIT_TRAIL` (if fine-grained auditing is enabled)

Related Topics

- *Oracle Database Vault Administrator's Guide*
- Performing Privilege Analysis to Identify Privilege Use
  Privilege analysis dynamically analyzes the privileges and roles that users use and do not use.

A.4 Guidelines for Securing Passwords

Oracle provides guidelines for securing passwords in a variety of situations.

When you create a user account, Oracle Database assigns a default password policy for that user. The password policy defines rules for how the password should be created, such as a minimum number of characters, when it expires, and so on. You can strengthen passwords by using password policies.

Follow these guidelines to further strengthen passwords:

1. **Choose passwords carefully.**

   In addition to the minimum requirements for creating passwords, follow these additional guidelines when you create or change passwords:

   - Make the password have a length of between 12 and 1024 bytes, and include both alphabetic characters and digits in the password.
   - Have the password contain at least one digit, one upper-case character, and one lower-case character.
   - Use mixed case characters and special characters in the password.
   - You can include multibyte characters in the password but not in the password of any common user or role.
   - Use the database character set for the password's characters, which can include the underscore (_), dollar ($), and number sign (#) characters.
   - You must enclose the following passwords in double-quotation marks:
     - Passwords containing multibyte characters.
     - Passwords starting with numbers or special characters and containing alphabetic characters (a–z, A–Z). For example:

       ```
       "123abc"

       "#abc"

       "123dc$"
       ```

     - Passwords containing any character other than alphabetic characters, numbers, and special characters. For example:

       ```
       "abc>"

       "abc@",

       " "
       ```

   - You do not need to specify the following passwords in double-quotation marks.
     - Passwords starting with an alphabetic character (a–z, A–Z) and containing numbers (0–9) or special characters ($, #, _). For example:

       ```
       abc123
       ```

```
ab23a

ab$#_
```

  - Passwords containing only numbers
  - Passwords containing only alphabetic characters (a–z, A–Z)
- Do not include double-quotation marks within the password.
- Do not use an actual word for the entire password.

2. **To create a longer, more complex password from a shorter, easier to remember password, create the password from the first letters of the words of an easy-to-remember sentence.**

   For example, "I usually work until 6:00 almost every day of the week" can be `Iuwu6aedotw`.

3. **Ensure that the password is sufficiently complex.**

   Oracle Database provides a password complexity verification routine, the PL/SQL script `utlpwdmg.sql`, that you can run to check whether or not passwords are sufficiently complex. Ideally, edit the `utlpwdmg.sql` script to provide stronger password protections.

4. **Remember that multibyte characters are not allowed in passwords for common users or roles.**
   For users who are local to a PDB, if you want to use multibyte characters in the password, then ensure that the database character set is configured as a multibyte character set so that the authentication will work properly.

   Be aware that because multibyte characters consume more bytes than single-byte characters, they tend to provide less entropy per byte. Because the maximum length of the password is limited to 1024 bytes, to help increase the amount of entropy in a password, Oracle recommends that you also include a number of single-byte characters in the password, even when multibyte characters are being used.

5. **Associate a password complexity function with the user profile or the default profile.**

   The `PASSWORD_VERIFY_FUNCTION` clause of the `CREATE PROFILE` and `ALTER PROFILE` statements associates a password complexity function with a user profile or the default profile. Password complexity functions ensure that users create strong passwords using guidelines that are specific to your site. Having a password complexity function also requires a user changing their own password (without the `ALTER USER` system privilege) to provide both the old and new passwords. You can create your own password complexity functions or use the password complexity functions that Oracle Database provides.

6. **Change default user passwords.**

   Oracle Database installs with a set of predefined, default user accounts. Security is most easily broken when a default database user account still has a default password *even after installation*. This is particularly true for the user account `SCOTT`, which is a well known account that may be vulnerable to intruders. In Oracle Database, default accounts are installed locked with the passwords expired, but if you have upgraded from a previous release, you may still have accounts that use default passwords.

   To find user accounts that have default passwords, query the `DBA_USERS_WITH_DEFPWD` data dictionary view.

7. **Change default passwords of administrative users.**

   You can use the same or different passwords for the `SYS`, `SYSTEM`, `SYSMAN`, and `DBSNMP` administrative accounts. Oracle recommends that you use different passwords for each. In any Oracle environment (production or test), assign strong, secure, and distinct passwords to these administrative accounts. If you use Database Configuration Assistant to create a

new database, then it requires you to enter passwords for the SYS and SYSTEM accounts, disallowing the default passwords CHANGE_ON_INSTALL and MANAGER.

Similarly, for production environments, do not use default passwords for administrative accounts, including SYSMAN and DBSNMP.

8. **Enforce password management.**

   Apply basic password management rules (such as password length, history, complexity, and so forth) to all user passwords. Oracle Database has password policies enabled for the default profile. Guideline 1 in this section lists these password policies.

   You can find information about user accounts by querying the DBA_USERS view. The PASSWORD column of the DBA_USERS view indicates whether the password is global, external, or null. The DBA_USERS view provides useful information such as the user account status, whether the account is locked, and password versions.

   Oracle also recommends, if possible, using Oracle strong authentication with network authentication services (such as Kerberos), token cards, smart cards, or X.509 certificates. These services provide strong authentication of users, and provide protection against unauthorized access to Oracle Database.

9. **Do not store user passwords in clear text in Oracle tables.**

   For better security, do not store passwords in clear text (that is, human readable) in Oracle tables. You can correct this problem by using a secure external password store to encrypt the password within an Oracle wallet. (An Oracle wallet is a secure software container that stores authentication and signing credentials.)

   When you create or modify a password for a user account, Oracle Database automatically creates a cryptographic hash or digest of the password. If you query the DBA_USERS view to find information about a user account, the data in the PASSWORD column indicates if the user password is global, external, or null. The DBA_USERS view also has a column called PASSWORD_VERSIONS, which lists the types of cryptographic hash that exist for the user's password (11G or 12C).

10. **Disable the HTTP verifier if the user is not going to be using either XDB authentication or HTTP Digest authentication.**
    The HTTP verifier is used only for XDB authentication and HTTP Digest authentication. If a user is not going to use XDB authentication or HTTP Digest authentication, then you can safely remove the HTTP verifier from the user's list of verifiers. To remove a user's HTTP verifier, run the following statement:

    ```
    ALTER USER username DIGEST DISABLE;
    ```

Related Topics

- Minimum Requirements for Passwords
  Oracle provides a set of minimum requirements for passwords.
- Configuring Password Protection
  You can secure user passwords in a variety of ways, such as controlling the password creation requirements or using password management policies.
- Ensuring Against Password Security Threats by Using the 12C Password Version
  The 12C password version enables users to create complex passwords that meet compliance standards.
- About Password Complexity Verification
  Complexity verification checks that each password is complex enough to protect against intruders who try to guess user passwords.

- Managing the Complexity of Passwords
  Oracle Database provides a set of functions that you can use to manage the complexity of passwords.
- Finding User Accounts That Have Default Passwords
  The `DBA_USERS_WITH_DEFPWD` data dictionary view can find user accounts that use default passwords.
- Managing the Secure External Password Store for Password Credentials
  The secure external password store (SEPS) is a client-side wallet that is used to store password credentials.

A.5 Securing Authentication for Oracle Database Microsoft Windows Installations

By default, the `SQLNET.NO_NTLM` parameter setting in the `sqlnet.ora` file on Microsoft Windows installations with `AUTHENTICATION_SERVICES=NTS` is `TRUE`.

If you upgrade from a previous release where the `SQLNET.NO_NTLM` parameter had not been set, then it defaults to `TRUE`.

You must include this setting on both the server and client, and this setting should be the same on both. Ideally, you should ensure that `SQLNET.NO_NTLM` is set to `TRUE`. However, if there is an authentication failure in `extproc`, a virtual account, or a local account on Windows, set the client `SQLNET.NO_NTLM` to `FALSE`, and then retry the login. If you change `SQLNET.NO_NTLM` on the server, then you must restart the database.

A.6 Guidelines for Securing Roles

Oracle provides guidelines for role management.

1. **Grant a role to users only if they need all privileges of the role.**

   Roles (groups of privileges) are useful for quickly and easily granting permissions to users. Although you can use Oracle-defined roles, you have more control and continuity if you create your own roles containing only the privileges pertaining to your requirements. Oracle may change or remove the privileges in an Oracle Database-defined role, as it has with the `CONNECT` role, which now has only the `CREATE SESSION` privilege. Formerly, this role had eight other privileges.

   Ensure that the roles you define contain only the privileges that reflect job responsibility. If your application users do not need all the privileges encompassed by an existing role, then apply a different set of roles that supply just the correct privileges. Alternatively, create and assign a more restricted role.

   For example, it is imperative to strictly limit the privileges of user `SCOTT`, because this is a well known account that may be vulnerable to intruders. Because the `CREATE DBLINK` privilege allows access from one database to another, drop its privilege for `SCOTT`. Then, drop the entire role for the user, because privileges acquired by means of a role cannot be dropped individually. Re-create your own role with only the privileges needed, and grant that new role to that user. Similarly, for better security, drop the `CREATE DBLINK` privilege from all users who do not require it.

2. **Do not grant user roles to application developers.**

   Roles are not meant to be used by application developers, because the privileges to access schema objects within stored programmatic constructs need to be granted directly.

Remember that roles are not enabled within stored procedures except for invoker's right procedures.

3. **Create and assign roles specific to each Oracle Database installation.**

   This principle enables the organization to retain detailed control of its roles and privileges. This also avoids the necessity to adjust if Oracle Database changes or removes Oracle Database-defined roles, as it has with `CONNECT`, which now has only the `CREATE SESSION` privilege. Formerly, it also had eight other privileges.

4. **For enterprise users, create global roles.**

   Global roles are managed by an enterprise directory service, such as Oracle Internet Directory.

Related Topics

- How Roles Work in PL/SQL Blocks
  Role behavior in a PL/SQL block is determined by the type of block and by definer's rights or invoker's rights.
- Authorizing a Global Role by an Enterprise Directory Service
  A global role enables a global user to be authorized only by an enterprise directory service.
- *Oracle Database Enterprise User Security Administrator's Guide*

A.7 Guidelines for Securing Data

Oracle provides guidelines for securing data on your system.

1. **Restrict operating system access.**

   Follow these guidelines:

   - Limit the number of operating system users.
   - Limit the privileges of the operating system accounts (administrative, root-privileged, or database administrative) on the Oracle Database host computer to the least privileges required for a user to perform necessary tasks.
   - Restrict the ability to modify the default file and directory permissions for the Oracle Database home (installation) directory or its contents. Even privileged operating system users and the Oracle owner should not modify these permissions, unless instructed otherwise by Oracle.
   - Restrict symbolic links. Ensure that when you provide a path or file to the database, neither the file nor any part of the path is modifiable by an untrusted user. The file and all components of the path should be owned by the database administrator or trusted account, such as *root*.

     This recommendation applies to all types of log files, trace files, external tables, BFILE data types, and so on.

2. **Encrypt sensitive data and all backup media that contains database files.**

   According to common regulatory compliance requirements, you must encrypt sensitive data such as credit card numbers and passwords. When you delete sensitive data from the database, encrypted data does not linger in data blocks, operating system files, or sectors on disk.

   In most cases, you may want to use Transparent Data Encryption to encrypt your sensitive data.

3. **For Oracle Automatic Storage Management (Oracle ASM) environments on Linux and UNIX systems, use Oracle ASM File Access Control to restrict access to the Oracle ASM disk groups.**

   If you use different operating system users and groups for Oracle Database installations, then you can configure Oracle ASM File Access Control to restrict the access to files in Oracle ASM disk groups to only authorized users. For example, a database administrator would only be able to access the data files for the databases that they manage. This administrator would not be able to see or overwrite the data files belonging (or used by) other databases.

   For more information about managing Oracle ASM File Access Control for disk groups and the various privileges that are required for multiple software owners, see *Oracle Automatic Storage Management Administrator's Guide*.

Related Topics

- Security Problems That Encryption Does Not Solve
  While there are many good reasons to encrypt data, there are many reasons not to encrypt data.
- *Oracle Database Advanced Security Guide*
- *Oracle Automatic Storage Management Administrator's Guide*
- *Oracle Automatic Storage Management Administrator's Guide*

A.8 Guidelines for Securing the ORACLE_LOADER Access Driver

Oracle provides guidelines to secure the `ORACLE_LOADER` access driver.

1. **Create a separate operating system directory to store the access driver preprocessors.** You (or the operating system manager) may need to create multiple directories if different Oracle Database users will run different preprocessors. If you want to prevent one set of users from using one preprocessor while allowing those users access to another preprocessor, then place the preprocessors in separate directories. If all the users need equal access, then you can place the preprocessors together in one directory. After you create these operating system directories, in SQL*Plus, you can create a directory object for each directory.
2. **Grant the operating system user ORACLE the correct operating system privileges to run the access driver preprocessor.** In addition, protect the preprocessor program from `WRITE` access by operating system users other than the user responsible for managing the preprocessor program.
3. **Grant the EXECUTE privilege to each user who will run the preprocessor program in the directory object.** Do not grant this user the `WRITE` privilege on the directory object. Never grant users both the `EXECUTE` and `WRITE` privilege for directory objects.
4. **Grant the WRITE privilege sparingly to anyone who will manage directory objects that contain preprocessors.** This prevents database users from accidentally or maliciously overwriting the preprocessor program.
5. **Create a separate operating system directory and directory object for any data files that are required for external tables.** Ensure that these are separate from the directory and directory object used by the access directory preprocessor.

   Work with the operating system manager to ensure that only the appropriate operating system users have access to this directory. Grant the `ORACLE` operating system user `READ`

access to any directory that has a directory object with `READ` privileges granted to database users. Similarly, grant the `ORACLE` operating system user `WRITE` access to any directory that has the `WRITE` privilege granted to database users.

6. **Create a separate operating system directory and directory object for any files that the access driver generates.** This includes log files, bad files, and discarded files. You and the operating system manager must ensure that this directory and directory object have the proper protections, similar to those described in Guideline 5. The database user may need to access these files when resolving problems in data files, so you and the operating system manager must determine a way for this user to read those files.
7. **Grant the CREATE ANY DIRECTORY and DROP ANY DIRECTORY privileges sparingly.** Users who have these privileges and users who have been granted the `DBA` role have full access to all directory objects.
8. **Consider auditing the DROP ANY DIRECTORY privilege.** You can create a unified audit policy to audit privileges.
9. **Consider auditing the directory object.** You can create a unified audit policy to audit objects.

Related Topics

- Auditing System Privileges
  You can use the `CREATE AUDIT POLICY` statement to audit system privileges.
- Auditing Object Actions
  You can use the `CREATE AUDIT POLICY` statement to audit object actions.
- *Oracle Database Utilities*

A.9 Guidelines for Securing a Database Installation and Configuration

Oracle provides guidelines to secure the database installation and configuration.

Changes were made to the default configuration of Oracle Database to make it more secure. The recommendations in this section augment the new, secure default configuration.

1. **Before you begin an Oracle Database installation on UNIX systems, ensure that the umask value is 022 for the Oracle owner account.**
2. **Install only what is required.**

   Options and Products: The Oracle Database CD pack contains products and options in addition to the database. Install additional products and options only as necessary. Use the Custom Installation feature to avoid installing unnecessary products, or perform a typical installation, and then deinstall options and products that are not required. There is no need to maintain additional products and options if they are not being used. They can always be properly installed, as required.

   Sample Schemas: Oracle Database provides sample schemas to provide a common platform for examples. If your database will be used in a production environment, then do not install the sample schema. If you have installed the sample schema on a test database, then before going to production, remove or relock the sample schema accounts.
3. **During installation, when you are prompted for a password, create a secure password.**

   Choose the password carefully, ensure that you change the default passwords, and change the default passwords of administrative users.

4. **Immediately after installation, lock and expire default user accounts.**

   For better security, you should lock and expire all default (predefined) user accounts.

Related Topics

- *Oracle Database Administrator's Reference for Linux and UNIX-Based Operating Systems*
- *Oracle Database Sample Schemas*
- Guidelines for Securing Passwords
  Oracle provides guidelines for securing passwords in a variety of situations.
- Guidelines for Securing User Accounts and Privileges
  Oracle provides guidelines to secure user accounts and privileges.

A.10 Guideline for Securing Multitenant PDBs from the Root in a Linux Environment

In Linux, you can securely compartmentalize PDBs to manage their resources in containers called nests.

A database instance that runs on a host must have isolation and resource management with respect to other databases and applications running in the same host. You can use security isolation to shield this database instance (even from the root), so that a security breach in any application does not affect the database instance.

To use this feature, you create a container, called a nest, around the pluggable database (PDB) that you want to protect. The nests are hierarchical. Each nest exists in isolation from other nests, and enables the nest administrator to manage isolation and resource settings for the PDB contained within the nest. Each nest provides the following features:

- Isolation of operating system resources, such as pid, mount, and network
- Resource management for resources such as CPU, memory, and network
- File system isolation, in which you can control the visibility for various system level entities in a nest
- Secure computing, to filter, enable, or disable required system calls at the nest level

Related Topics

- *Oracle Multitenant Administrator's Guide*

A.11 Guidelines for Securing the Network

Security for network communications is improved by using client, listener, and network guidelines to ensure thorough protection.

- Client Connection Security
  Authenticating clients stringently, configuring encryption for the connection, and using strong authentication strengthens client connections.
- Network Connection Security
  Protecting the network and its traffic from inappropriate access or modification is the essence of network security.
- Transport Layer Security Connection Security
  Oracle provides guidelines for securing Transport Layer Security (TLS).

A.11.1 Client Connection Security

Authenticating clients stringently, configuring encryption for the connection, and using strong authentication strengthens client connections.

Because authenticating client computers is problematic, typically, user authentication is performed instead. This approach avoids client system issues that include falsified IP addresses, hacked operating systems or applications, and falsified or stolen client system identities.

Nevertheless, the following guidelines improve the security of client connections:

1. **Configure the connection to use encryption.**

   Oracle native network encryption makes eavesdropping difficult.

2. **Set up strong authentication.**

   You can use Kerberos authentication and public key infrastructure (PKI).

3. **In an Oracle Data Guard environment, set the `ADG_ACCOUNT_INFO_TRACKING` initialization parameter.**

   The `ADG_ACCOUNT_INFO_TRACKING` parameter controls login attempts on Oracle Active Data Guard standby databases. It provides more security against login attacks across an Oracle Database production environment and all Active Data Guard standby databases. Use one of the following settings:

   - `LOCAL` (default) enforces the existing behavior, which maintains a local copy of user account information in the standby database's in-memory view. This setting only tracks login failures locally on a per-database basis. It denies the login when the maximum of failed logins is reached.
   - `GLOBAL` increases the security of logins by maintaining a single global copy of user account information across all Data Guard primary and standby databases. Login failures across all databases in the Data Guard environment count toward the maximum count. When this count is reached, then logins anywhere are denied access.

Related Topics

- Configuring Kerberos Authentication
  Kerberos is a trusted third-party authentication system that relies on shared secrets and presumes that the third party is secure.
- *Oracle Database Reference*

A.11.2 Network Connection Security

Protecting the network and its traffic from inappropriate access or modification is the essence of network security.

You should consider all paths the data travels, and assess the threats on each path and node. Then, take steps to lessen or eliminate those threats and the consequences of a security breach. In addition, monitor and audit to detect either increased threat levels or penetration attempts.

To manage network connections, you can use Oracle Net Manager. For more information about Net Manager, see *Oracle Database Net Services Administrator's Guide*.

The following practices improve network security:

1. **Use Transport Layer Security (TLS) when administering the listener.**

   TLS can protect the messages sent and received by you or by applications and servers, supporting secure authentication, authorization, and messaging through certificates and, if necessary, encryption.

2. **Prevent online administration by requiring the administrator to have the write privilege on the listener password and on the listener.ora file on the server.**

   a. Add or alter this line in the `listener.ora` file:

   ```
   ADMIN_RESTRICTIONS_LISTENER=ON
   ```

   b. Use `RELOAD` to reload the configuration.

   c. Use TLS when administering the listener by making the TCPS protocol the first entry in the address list, as follows:

   ```
   LISTENER=
     (DESCRIPTION=
       (ADDRESS_LIST=
         (ADDRESS=
           (PROTOCOL=tcps)
           (HOST = sales.us.example.com)
           (PORT = 8281)))
   ```

   To administer the listener remotely, you define the listener in the `listener.ora` file on the client computer. For example, to access listener USER281 remotely, use the following configuration:

   ```
   user281 =
     (DESCRIPTION =
       (ADDRESS =
         (PROTOCOL = tcps)
         (HOST = sales.us.example.com)
         (PORT = 8281))
       )
     )
   ```

3. **Do not set the listener password.**

   Ensure that the password has not been set in the `listener.ora` file. The local operating system authentication will secure the listener administration. The remote listener administration is disabled when the password has not been set. This prevents brute force attacks of the listener password.

   The listener password has been deprecated in this release. It will not be supported in the next release of Oracle Database.

4. **When a host computer has multiple IP addresses associated with multiple network interface controller (NIC) cards, configure the listener to the specific IP address.**

   This allows the listener to listen on all the IP addresses. You can restrict the listener to listen on a specific IP address. Oracle recommends that you specify the specific IP addresses on these types of computers, rather than allowing the listener to listen on all IP addresses. Restricting the listener to specific IP addresses helps to prevent an intruder from stealing a TCP end point from under the listener process.

5. **Restrict the privileges of the listener, so that it cannot read or write files in the database or the Oracle server address space.**

   This restriction prevents external procedure agents spawned by the listener (or procedures run by an agent) from inheriting the ability to perform read or write operations. The owner of this separate listener process should not be the owner that installed Oracle Database or runs the Oracle Database instance (such as `ORACLE`, the default owner).

For more information about configuring external procedures in the listener, see *Oracle Database Net Services Administrator's Guide*.

6. **Use encryption to secure the data in flight.**

   Strong authentication will help to protect network data encryption.

7. **Use a firewall.**

   Appropriately placed and configured firewalls can prevent outside access to your databases.

   - Keep the database server behind a firewall. Oracle Database network infrastructure, Oracle Net Services (formerly known as SQL*Net), provides support for a variety of firewalls from various vendors. Supported proxy-enabled firewalls include Gauntlet from Network Associates and Raptor from Axent. Supported packet-filtering firewalls include PIX Firewall from Cisco, and supported stateful inspection firewalls (more sophisticated packet-filtered firewalls) include Firewall-1 from CheckPoint.
   - Ensure that the firewall is placed outside the network to be protected.
   - Configure the firewall to accept only those protocols, applications, or client/server sources that you know are safe.
   - Use a product such as Net8 and Oracle Connection Manager to manage multiplex multiple client network sessions through a single network connection to the database. It can filter on source, destination, and host name. This product enables you to ensure that connections are accepted only from physically secure terminals or from application Web servers with known IP addresses. (Filtering on IP address alone is not enough for authentication, because it can be falsified.)

8. **Prevent unauthorized administration of the Oracle listener.**

   For more information about the listener, see *Oracle Database Net Services Administrator's Guide*.

9. **Check network IP addresses.**

   Use the Oracle Net *valid node checking* security feature to allow or deny access to Oracle server processes from network clients with specified IP addresses. To use this feature, set the following `sqlnet.ora` configuration file parameters:

   ```
   tcp.validnode_checking = YES

   tcp.excluded_nodes = {list of IP addresses}

   tcp.invited_nodes = {list of IP addresses}
   ```

   The `tcp.validnode_checking` parameter enables the feature. The `tcp.excluded_nodes` and `tcp.invited_nodes` parameters deny and enable specific client IP addresses from making connections to the Oracle listener. This helps to prevent potential Denial of Service attacks.

10. **Set Oracle Connection Manager parameters to prevent denial-of-service attacks.** The following parameters in the Oracle Connection Manager `cman.ora` configuration file set a limit on the number of new connections that are allowed from an IP address in the specified unit of time:

    - `IP_RATE_COUNT`: Specifies the number of new connections allowed from an IP address in the specified time interval.
    - `IP_RATE_INTERVAL`: Specifies the time interval, in seconds, for which `IP_RATE_COUNT` connections are accepted from the IP address.

- `IP_RATE_BLOCK`: Specifies the duration, in minutes, for which the IP address is blocked after exceeding the specified IP rate limit.

See *Oracle Database Net Services Administrator's Guide*.

11. **Encrypt network traffic.**

    If possible, use Oracle native network data encryption to encrypt network traffic among clients, databases, and application servers.

12. **Secure the host operating system (the system on which Oracle Database is installed).**

    Secure the host operating system by disabling all unnecessary operating system services. Both UNIX and Windows provide a variety of operating system services, most of which are not necessary for typical deployments. These services include FTP, TFTP, TELNET, and so forth. Be sure to close both the UDP and TCP ports for each service that is being disabled. Disabling one type of port and not the other does not make the operating system more secure.

13. **Configure database link communication protocol.**

    To specify the protocols over which the database link communication takes place, set the `OUTBOUND_DBLINK_PROTOCOLS` initialization parameter to one of the following settings:

    - `ALL` (default) enables all net protocols to be used for the database links.
    - *`comma-separated_list_of_protocols`* can be set `TPC`, `TCPS`, or `IPC`. For example, for a single protocol:

      ```
      ALTER SYSTEM SET OUTBOUND_DBLINK_PROTOCOLS=TCPS;
      ```

      For multiple protocols:

      ```
      ALTER SYSTEM SET OUTBOUND_DBLINK_PROTOCOLS=TCP,TCPS,IPC;
      ```

    - `NONE` disables any database link communication.

14. **If necessary, disable LDAP lookup for global database links.**

    Set the `ALLOW_GLOBAL_DBLINKS` initialization parameter to enable or disable LDAP lookup for global database links. Settings are as follows:

    - `ON` enables LDAP lookup for global database links.
    - `OFF` (default) disables LDAP lookup for global database links.

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- Configuring PKI Certificate Authentication
  You can configure Oracle Database to use PKI certificates for end-user authentication.
- *Oracle Database Net Services Administrator's Guide*
- Introduction to Strong Authentication
  Strong authentication supports tools such as Transport Layer Security (TLS) to verify the identities of users who log in to the database.
- *Oracle Database Net Services Administrator's Guide*
- Configuring Oracle Database Native Network Encryption and Data Integrity
  You can configure native Oracle Net Services data encryption and data integrity for both servers and clients.

A.11.3 Transport Layer Security Connection Security

Oracle provides guidelines for securing Transport Layer Security (TLS).

Transport Layer Security (TLS) is the Internet standard protocol for secure communication, providing mechanisms for data integrity and data encryption. These mechanisms can protect the messages sent and received by you or by applications and servers, supporting secure authentication, authorization, and messaging through certificates and, if necessary, encryption. Good security practices maximize protection and minimize gaps or disclosures that threaten security.

1. **Ensure that configuration files (for example, for clients and listeners) use the correct port for TLS, which is the port configured upon installation.**

   You can run HTTPS on any port, but the standards specify port 443, where any HTTPS-compliant browser looks by default. The port can also be specified in the URL, for example:

   ```
   https://secure.example.com:4445/
   ```

   If a firewall is in use, then it too must use the same ports for secure (TLS) communication.

2. **Ensure that TCPS is specified as the PROTOCOL in the ADDRESS parameter in the tnsnames.ora file (typically on the client or in the LDAP directory).**

   An identical specification must appear in the `listener.ora` file (typically in the `$ORACLE_HOME/network/admin` directory).

3. **Ensure that the TLS mode is consistent for both ends of every communication. For example, the database (on one side) and the user or application (on the other) must have the same TLS mode.**

   The mode can specify either client or server authentication (one-way), both client and server authentication (two-way), or no authentication.

4. **Ensure that the server supports the client cipher suites and the certificate key algorithm in use.**

5. **Enable DN matching for both the server and client, to prevent the server from falsifying its identity to the client during connections.**

   This setting ensures that the server identity is correct by matching its global database name against the DN from the server certificate.

   You can enable DN matching in the `tnsnames.ora` file. For example:

   ```
   set:SSL_SERVER_CERT_DN="cn=finance,cn=OracleContext,c=us,o=example"
   ```

   Otherwise, a client application would not check the server certificate, which could allow the server to falsify its identity.

6. **Do not remove the encryption from your RSA private key inside your server.key file, which requires that you enter your pass phrase to read and parse this file.**

> **Note:**
>
> A server without TLS does not require a pass phrase.

If you decide your server is secure enough, you could remove the encryption from the RSA private key while preserving the original file. This enables system boot scripts to start the database server, because no pass phrase is needed. Ideally, restrict permissions to the root user only, and have the Web server start as `root`, but then log on as another user. Otherwise, anyone who gets this key can impersonate you on the Internet, or decrypt the data that was sent to the server.

Related Topics

- Configuring PKI Certificate Authentication
  You can configure Oracle Database to use PKI certificates for end-user authentication.
- *Oracle Database Net Services Reference*

A.12 Guideline for Securing External Procedures

The `ENFORCE_CREDENTIAL` environment variable controls how an `extproc` process authenticates user credentials and callout functions.

You can specify this variable in the `extproc.ora` file. Before modifying this variable, review your site's security requirements for the handling of external libraries. For maximum security, set the `ENFORCE_CREDENTIAL` variable to `TRUE`. The default setting is `FALSE`.

Related Topics

- Securing External Procedures
  An external procedure is stored in a `.dll` or an `.so` file, separately from the database, and can be through a credential authentication.

A.13 Guidelines for Auditing

Oracle provides guidelines for auditing.

- Manageability of Audited Information
  Although auditing is relatively inexpensive, limit the number of audited events as much as possible.
- Audits of Typical Database Activity
  Oracle provides guidelines for when you must gather historical information about particular database activities.
- Audits of Suspicious Database Activity
  Oracle provides guidelines for when you audit to monitor suspicious database activity.
- Audits of Sensitive Data
  Oracle recommends that you include the `ACTIONS ALL` clause when you create unified audit policies on sensitive objects.
- Recommended Audit Settings
  Oracle provides predefined policies that contain recommended audit settings that apply to most sites.
- Best Practices for Querying the UNIFIED_AUDIT_TRAIL Data Dictionary View
  To get the best results from querying the `UNIFIED_AUDIT_TRAIL` data dictionary view, you should follow these guidelines.

A.13.1 Manageability of Audited Information

Although auditing is relatively inexpensive, limit the number of audited events as much as possible.

This minimizes the performance impact on the execution of audited statements and the size of the audit trail, making it easier to analyze and understand.

Follow these guidelines when devising an auditing strategy:

1. **Evaluate your reason for auditing.**

   After you have a clear understanding of the reasons for auditing, you can devise an appropriate auditing strategy and avoid unnecessary auditing.

   For example, suppose you are auditing to investigate suspicious database activity. This information by itself is not specific enough. What types of suspicious database activity do you suspect or have you noticed? A more focused auditing strategy might be to audit unauthorized deletions from arbitrary tables in the database. This purpose narrows the type of action being audited and the type of object being affected by the suspicious activity.

2. **Audit knowledgeably.**

   Audit the minimum number of statements, users, or objects required to get the targeted information. This prevents unnecessary audit information from cluttering the meaningful information and using valuable space in the `SYSTEM` tablespace. Balance your need to gather sufficient security information with your ability to store and process it.

   For example, if you are auditing to gather information about database activity, then determine exactly what types of activities you want to track, audit only the activities of interest, and audit only for the amount of time necessary to gather the information that you want. As another example, do not audit *objects* if you are only interested in logical I/O information for each session.

3. **Before you implement an auditing strategy, consult your legal department.**

   You should have the legal department of your organization review your audit strategy. Because your auditing will monitor other users in your organization, you must ensure that you are correctly following the compliance and corporate policy of your site.

A.13.2 Audits of Typical Database Activity

Oracle provides guidelines for when you must gather historical information about particular database activities.

1. **Audit only pertinent actions.**

   At a minimum, audit user access, the use of system privileges, and changes to the database schema structure. To avoid cluttering meaningful information with useless audit records and reduce the amount of audit trail administration, only audit the targeted database activities. Remember also that auditing too much can affect database performance.

   For example, auditing changes to all tables in a database produces far too many audit trail records and can slow down database performance. However, auditing changes to critical tables, such as salaries in a Human Resources table, is useful.

   You can audit specific actions by using fine-grained auditing.

2. **Archive audit records and purge the audit trail.**

After you collect the required information, archive the audit records of interest and then purge the audit trail of this information.

3. **Remember your company's privacy considerations.**

   Privacy regulations often lead to additional business privacy policies. Most privacy laws require businesses to monitor access to personally identifiable information (PII), and monitoring is implemented by auditing. A business-level privacy policy should address all relevant aspects of data access and user accountability, including technical, legal, and company policy concerns.

4. **Check the Oracle Database log files for additional audit information.**

   The log files generated by Oracle Database contain useful information that you can use when auditing a database. For example, an Oracle database creates an alert file to record `STARTUP` and `SHUTDOWN` operations, and structural changes such as adding data files to the database.

   For example, if you want to audit committed or rolled back transactions, you can use the redo log files.

5. **To reduce the size of the audit trail and recursive SQL statements, audit only top-level statements.**
   If you have concerns that the unified audit policy that you create will generate a very large number of records, then include the `ONLY TOPLEVEL` clause in the `CREATE AUDIT POLICY` statement. For example, an audit of the `DBMS_STATS.GATHER_DATABASE_STATS` SQL statement can generate thousands of audit records. You can audit top-level statements from all users, including user `SYS`.

Related Topics

- Value-Based Auditing with Fine-Grained Audit Policies
  Fine-grained auditing enables you to perform value-based auditing to audit access to certain rows based on values in specific columns.
- Archiving the Audit Trail
  To maintain the integrity and reliability of audit data, keep only minimal required audit data locally in the database.
- Purging Audit Trail Records
  The `DBMS_AUDIT_MGMT` PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

A.13.3 Audits of Suspicious Database Activity

Oracle provides guidelines for when you audit to monitor suspicious database activity.

1. **First audit generally, and then specifically.**

   When you start to audit for suspicious database activity, often not much information is available to target specific users or schema objects. Therefore, audit generally first, that is, by using the unified audit policies. You can audit SQL statements, schema objects, privileges, and so on.

   After you have recorded and analyzed the preliminary audit information, alter your audit policies to audit specific actions and privileges. You can add conditions to your policies to exclude unnecessary audit records. You an also use the `EXCEPT` clause in the `AUDIT POLICY` statement to exclude specific users who do not need to be audited.

   You can use fine-grained auditing to audit specific actions, such as when and where a user logged in to a specific database instance.

Continue this process until you have gathered enough evidence to draw conclusions about the origin of the suspicious database activity.

2. **Audit common suspicious activities.**

   Common suspicious activities are as follows:

   - Users who access the database during unusual hours
   - Multiple failed user login attempts
   - Login attempts by non-existent users

   In addition, be aware that sensitive data, such as credit card numbers, can appear in the audit trail columns, such as SQL text when used in the SQL query. You should also monitor users who share accounts or multiple users who are logging in from the same IP address. You can query the `UNIFIED_AUDIT_TRAIL` data dictionary view to find this kind of activity. For a very granular approach, create fine-grained audit policies.

Related Topics

- Provisioning Audit Policies
  Oracle Database provides a variety of ways for you to audit activities.
- Creating Custom Unified Audit Policies
  Oracle Database provides the flexibility to create and manage custom unified audit policies for your specialized needs.
- Value-Based Auditing with Fine-Grained Audit Policies
  Fine-grained auditing enables you to perform value-based auditing to audit access to certain rows based on values in specific columns.

A.13.4 Audits of Sensitive Data

Oracle recommends that you include the `ACTIONS ALL` clause when you create unified audit policies on sensitive objects.

Including this clause ensures the generation of audit record for both direct access and indirect access of these sensitive objects. Only use `ACTIONS ALL` for the audit of sensitive objects.

Related Topics

- Example: Auditing All Actions on a Table
  The `CREATE AUDIT POLICY` statement can audit all actions on a table.

A.13.5 Recommended Audit Settings

Oracle provides predefined policies that contain recommended audit settings that apply to most sites.

For example:

- `ORA_SECURECONFIG` audits the same default audit settings from Oracle Database Release 11*g*. It tracks the use of a number of privileges such as `ALTER ANY TABLE`, `GRANT ANY PRIVILEGE`, and `CREATE USER`. The actions that it tracks include `ALTER USER`, `CREATE ROLE`, `LOGON`, and other commonly performed activities. This policy is enabled by default only when the database is created in Oracle Database Release 12*c*.
- `ORA_DATABASE_PARAMETER` audits commonly used Oracle Database parameter settings: `ALTER DATABASE`, `ALTER SYSTEM`, and `CREATE SPFILE`. By default, this policy is not enabled.

- `ORA_ACCOUNT_MGMT` audits the commonly used user account and privilege settings: `CREATE USER`, `ALTER USER`, `DROP USER`, `CREATE ROLE`, `DROP ROLE`,`ALTER ROLE`, `SET ROLE`, `GRANT`, and `REVOKE`. By default, this policy is not enabled.

Related Topics

- Auditing Activities with the Predefined Unified Audit Policies
  Oracle Database provides predefined unified audit policies that cover commonly used security-relevant audit settings.

A.13.6 Best Practices for Querying the UNIFIED_AUDIT_TRAIL Data Dictionary View

To get the best results from querying the `UNIFIED_AUDIT_TRAIL` data dictionary view, you should follow these guidelines.

1. **Ensure the statistics of unified audit internal table are up to date.**
   Run the `DBMS_STATS.GATHER_TABLE_STATS` procedure on the `AUD$UNIFIED` table in the `AUDSYS` schema to ensure that the unified audit table statistics are updated before you query the `UNIFIED_AUDIT_TRAIL` data dictionary view.
2. **Load the unified audit records that were written to operating system spillover files.**
   You can do this either explicitly or by configuring an Oracle Scheduler job, using the `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` procedure.
3. **When the number of records in the unified audit trail reaches a significantly large number (for example, a million), then initiate the proper archiving and purging mechanisms.**
   Archiving and purging the unified audit trial reduces the amount of data that otherwise could grow and cause read performance problems. Oracle recommends that you configure standard purging policies. The purging policies that you create will depend on the rate of audit records that are generated on your system. Frequent purges are required for high audit record generation rates.
4. **Move the unified audit trail to a custom tablespace.**
   Using a custom tablespace enables you to better manage audit data and reduces the impact on other objects in the `SYSAUX` tablespace. By default, the unified audit trail records are written to the `SYSAUX` tablespace. To use a different tablespace, run the `DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_LOCATION` procedure.
5. **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 of audited events in the UTC timezone. Including this column in the query helps to achieve the partition pruning, and thus improves read performance of the `UNIFIED_AUDIT_TRAIL` view.

Related Topics

- Moving Operating System Audit Records into the Unified Audit Trail
  Audit records that have been written to the spillover audit files can be moved to the unified audit trail database table.
- Archiving the Audit Trail
  To maintain the integrity and reliability of audit data, keep only minimal required audit data locally in the database.
- Purging Audit Trail Records
  The `DBMS_AUDIT_MGMT` PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

A.14 Addressing the CONNECT Role Change

The CONNECT role, introduced with Oracle Database version 7, added new and robust support for database roles.

- Why Was the CONNECT Role Changed?
  The CONNECT role is used in sample code, applications, documentation, and technical papers.
- How the CONNNECT Role Change Affects Applications
  The CONNECT role changes can be seen in database upgrades, account provisioning, and installation of applications using new databases.
- How the CONNECT Role Change Affects Users
  The change to the CONNECT role affects general users, application developers, and client/server applications differently.
- Approaches to Addressing the CONNECT Role Change
  Oracle recommends three approaches to address the impact of the CONNECT role change.

A.14.1 Why Was the CONNECT Role Changed?

The CONNECT role is used in sample code, applications, documentation, and technical papers.

In Oracle Database 10g release 2 (10.2), the CONNECT role was changed. If you are upgrading from a release earlier than Oracle Database 10.2 to the current release, then you should be aware of how the CONNECT role has changed in the most recent release.

The CONNECT role was originally established a special set of privileges. These privileges were as follows: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW.

Beginning in Oracle Database 10g release 2, the CONNECT role has only the CREATE SESSION privilege, all other privileges are removed. Starting with Oracle Database 12c release 1, the CONNECT role had the CREATE SESSION and SET CONTAINER privileges.

Although the CONNECT role was frequently used to provision new accounts in Oracle Database, connecting to the database does not require all those privileges. Making this change enables you to enforce good security practices more easily.

Each user should have only the privileges needed to perform their tasks, an idea called the principle of least privilege. Least privilege mitigates risk by limiting privileges, so that it remains easy to do what is needed while concurrently reducing the ability to do inappropriate things, either inadvertently or maliciously.

A.14.2 How the CONNNECT Role Change Affects Applications

The CONNECT role changes can be seen in database upgrades, account provisioning, and installation of applications using new databases.

- How the CONNECT Role Change Affects Database Upgrades
  You should be aware of how the CONNECT role affects database upgrades.
- How the CONNECT Role Change Affects Account Provisioning
  You should be aware of how the CONNECT role affects accounts provisioning.

- How the CONNECT Role Change Affects Applications Using New Databases
  You should be aware of how the CONNECT role affects applications that use new databases.

A.14.2.1 How the CONNECT Role Change Affects Database Upgrades

You should be aware of how the CONNECT role affects database upgrades.

Upgrading your existing Oracle database to Oracle Database 10*g* Release 2 (10.2) automatically changes the CONNECT role to have only the CREATE SESSION privilege.

Most applications are not affected because the applications objects already exist: no new tables, views, sequences, synonyms, clusters, or database links need to be created.

Applications that create tables, views, sequences, synonyms, clusters, or database links, or that use the ALTER SESSION command dynamically, may fail due to insufficient privileges.

A.14.2.2 How the CONNECT Role Change Affects Account Provisioning

You should be aware of how the CONNECT role affects accounts provisioning.

If your application or DBA grants the CONNECT role as part of the account provisioning process, then only CREATE SESSION privileges are included. Any additional privileges must be granted either directly or through another role.

This issue can be addressed by creating a new customized database role.

Related Topics

- Approaches to Addressing the CONNECT Role Change
  Oracle recommends three approaches to address the impact of the CONNECT role change.

A.14.2.3 How the CONNECT Role Change Affects Applications Using New Databases

You should be aware of how the CONNECT role affects applications that use new databases.

New databases created using the Oracle Database 10*g* Release 2 (10.2) Utility (DBCA), or using database creation templates generated from DBCA, define the CONNECT role with only the CREATE SESSION privilege.

Installing an application to use a new database may fail if the database schema used for the application is granted privileges solely through the CONNECT role.

A.14.3 How the CONNECT Role Change Affects Users

The change to the CONNECT role affects general users, application developers, and client/server applications differently.

- How the CONNECT Role Change Affects General Users
  You should be aware of how the CONNECT role affects general users.
- How the CONNECT Role Change Affects Application Developers
  You should be aware of how the CONNECT role affects application developers.
- How the CONNECT Role Change Affects Client Server Applications
  You should be aware of how the CONNECT role affects client server applications.

A.14.3.1 How the CONNECT Role Change Affects General Users

You should be aware of how the CONNECT role affects general users.

The new CONNECT role supplies only the CREATE SESSION privilege. Users who connect to the database to use an application are not affected, because the CONNECT role still has the CREATE SESSION privilege.

However, appropriate privileges will not be present for a certain set of users if they are provisioned solely with the CONNECT role. These are users who create tables, views, sequences, synonyms, clusters, or database links, or use the ALTER SESSION command. The privileges they need are no longer provided with the CONNECT role. To authorize the additional privileges needed, the database administrator must create and apply additional roles for the appropriate privileges, or grant them directly to the users who need them.

Note that the ALTER SESSION privilege is required for setting events. Few database users should require the ALTER SESSION privilege.

The ALTER SESSION privilege is *not* required for other alter session commands.

A.14.3.2 How the CONNECT Role Change Affects Application Developers

You should be aware of how the CONNECT role affects application developers.

Application developers provisioned solely with the CONNECT role do not have appropriate privileges to create tables, views, sequences, synonyms, clusters, or database links, nor to use the ALTER SESSION statement.

You must either create and apply additional roles for the appropriate privileges, or grant them directly to the application developers who need them.

A.14.3.3 How the CONNECT Role Change Affects Client Server Applications

You should be aware of how the CONNECT role affects client server applications.

Most client/server applications that use dedicated user accounts will not be affected by this change.

However, applications that create private synonyms or temporary tables using dynamic SQL in the user schema during account provisioning or run-time operations will be affected. They will require additional roles or grants to acquire the system privileges appropriate to their activities.

A.14.4 Approaches to Addressing the CONNECT Role Change

Oracle recommends three approaches to address the impact of the CONNECT role change.

- Creating a New Database Role
  The privileges removed from the CONNECT role can be managed by creating a new database role.
- Restoring the CONNECT Privilege
  The rstrconn.sql script restores the CONNECT privileges.
- Data Dictionary View to Show CONNECT Grantees
  The DBA_CONNECT_ROLE_GRANTEES data dictionary view enables administrators who continue using the old CONNECT role to see which users have that role.

- Least Privilege Analysis Studies
  Oracle partners and application providers should conduct a least privilege analysis so that they can deliver more secure products to their Oracle customers.

A.14.4.1 Creating a New Database Role

The privileges removed from the `CONNECT` role can be managed by creating a new database role.

1. Connect to the upgraded Oracle database and create a new database role.

   The following example uses a role called `my_app_developer`.

   ```
   CREATE ROLE my_app_developer;
   GRANT CREATE TABLE, CREATE VIEW, CREATE SEQUENCE, CREATE SYNONYM, CREATE CLUSTER,
   CREATE DATABASE LINK, ALTER SESSION TO my_app_developer;
   ```

2. Determine which users or database roles have the `CONNECT` role, and grant the new role to these users or roles.

   ```
   SELECT USER$.NAME, ADMIN_OPTION, DEFAULT_ROLE
    FROM USER$, SYSAUTH$, DBA_ROLE_PRIVS
    WHERE PRIVILEGE# =
    (SELECT USER# FROM USER$ WHERE NAME = 'CONNECT')
    AND USER$.USER# = GRANTEE#
    AND GRANTEE = USER$.NAME
    AND GRANTED_ROLE = 'CONNECT';

   NAME                           ADMIN_OPTI DEF
   ------------------------------ ---------- ---
   R1                             YES        YES
   R2                             NO         YES

   GRANT my_app_developer TO R1 WITH ADMIN OPTION;
   GRANT my_app_developer TO R2;
   ```

3. Determine the privileges that users require by creating a privilege analysis policy.

   The information that you gather can then be analyzed and used to create additional database roles with finer granularity. Privileges that are not used can then be revoked for specific users.

   For example:

   ```
   BEGIN
    DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
     name          => 'my_app_dev_role_pol',
     description   => 'Captures my_app_developer role use',
     type          => DBMS_PRIVILEGE_CAPTURE.G_ROLE,
     roles         => role_name_list('my_app_developer');
   END;
   /
   EXEC DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ('my_app_dev_role_pol');
   ```

4. After a period of time, disable the privilege analysis policy and then generate a report.

   ```
   EXEC DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ('my_app_dev_role_pol');

   EXEC DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT ('my_app_dev_role_pol');
   ```

5. After you generate the report, query the privilege analysis data dictionary views.

   For example:

```
SELECT USERNAME, SYS_PRIV, OBJECT_OWNER, OBJECT_NAME FROM DBA_USED_PRIVS;
```

Related Topics

- Performing Privilege Analysis to Identify Privilege Use
  Privilege analysis dynamically analyzes the privileges and roles that users use and do not use.

A.14.4.2 Restoring the CONNECT Privilege

The `rstrconn.sql` script restores the `CONNECT` privileges.

After a database upgrade or new database creation, you can use this script to grant the privileges that were removed from the `CONNECT` role in Oracle Database 10*g* release 2 (10.2). If you use this approach, then you should revoke privileges that are not used from users who do not need them.

To restore the `CONNECT` privilege:

1. Run the `rstrconn.sql` script, which is in the `$ORACLE_HOME/rdbms/admin` directory.

   ```
   @$ORACLE_HOME/rdbm_admin/rstrconn.sql
   ```

2. Monitor the privileges that are used.

   For example:

   ```
   CREATE AUDIT POLICY connect_priv_pol
     PRIVILEGES AUDIT CREATE TABLE, CREATE SEQUENCE, CREATE SYNONYM, CREATE DATABASE
   LINK, CREATE CLUSTER, CREATE VIEW, ALTER SESSION;

   AUDIT POLICY connect_priv_pol BY psmith;
   ```

3. Periodically, monitor database privilege usage.

   For example:

   ```
   SELECT USERID, NAME FROM AUD$, SYSTEM_PRIVILEGE_MAP WHERE - PRIV$USED = PRIVILEGE;

   USERID                         NAME
   ------------------------------ ----------------
   ACME                           CREATE TABLE
   ACME                           CREATE SEQUENCE
   ACME                           CREATE TABLE
   ACME                           ALTER SESSION
   APPS                           CREATE TABLE
   APPS                           CREATE TABLE
   APPS                           CREATE TABLE
   APPS                           CREATE TABLE
   8 rows selected.
   ```

A.14.4.3 Data Dictionary View to Show CONNECT Grantees

The `DBA_CONNECT_ROLE_GRANTEES` data dictionary view enables administrators who continue using the old `CONNECT` role to see which users have that role.

Table A-1 shows the columns in the `DBA_CONNECT_ROLE_GRANTEES` view.

Table A-1 Columns and Contents for DBA_CONNECT_ROLE_GRANTEES

Column	Datatype	NULL	Description
GRANTEE	VARCHAR2(128)	NULL	User granted the CONNECT role
PATH_OF_CONNECT_ROLE_GRANT	VARCHAR2(4000	NULL	Role (or nested roles) by which the user is granted CONNECT
ADMIN_OPT	VARCHAR2(3)	NULL	YES if user has the ADMIN option on CONNECT; otherwise, NO

A.14.4.4 Least Privilege Analysis Studies

Oracle partners and application providers should conduct a least privilege analysis so that they can deliver more secure products to their Oracle customers.

The principle of least privilege mitigates risk by limiting privileges to the minimum set required to perform a given function.

For each class of users that the analysis shows need the same set of privileges, create a role with only those privileges. Remove all other privileges from those users, and assign that role to those users. As needs change, you can grant additional privileges, either directly or through these new roles, or create new roles to meet new needs. This approach helps to ensure that inappropriate privileges have been limited, thereby reducing the risk of inadvertent or malicious harm.

You can create privilege analysis policies that show the use of privileges by database users. The policies capture this information and make it available in data dictionary views. Based on these reports, you can determine who should have access to your data.

Related Topics

- Performing Privilege Analysis to Identify Privilege Use
  Privilege analysis dynamically analyzes the privileges and roles that users use and do not use.