5
Security

This chapter explains some fundamentals of designing security into the database and database applications.

Topics:

5.1 Enabling User Access with Grants, Roles, and Least Privilege

This topic explains how you can grant privileges and roles to users to restrict access to data. It also explains the importance of the concept of least privilege, introduces secure application roles as a way to automatically filter out users who attempt to log in to your applications.

A user **privilege** is the right to perform an action, such as updating or deleting data. You can grant users privileges to perform these actions. A **role** is named collection of privileges that are grouped together, usually to enable users to perform a set of tasks related to their jobs. For example, a role called `clerk` can enable clerks to do things like create, update, and delete files. The `clerk_mgr` role can include the `clerk` role, plus some additional privileges such as approving the clerks' expense reports or managing their performance appraisals.

When you grant privileges to users, apply the principle of least privilege: *Grant users only the privileges that they need*. If possible, do not directly grant the user a privilege. Instead, create a role that defines the set of privileges the user needs and then grant the user this role. For example, grant user `fred` the `CREATE SESSION` privilege so that he can log in to a database session. But for the privileges that he needs for his job, such as the `UPDATE TABLE` privilege, grant him a role that has those privileges.

You can design your applications to automatically grant a role to the user who is trying to log in, provided the user meets criteria that you specify. To do so, you create a **secure application role**, which is a role that is associated with a PL/SQL procedure (or PL/SQL package that contains multiple procedures). The procedure validates the user: if the user fails the validation, then the user cannot log in. If the user passes the validation, then the procedure grants the user a role so that they can use the application. The user has this role only while they are logged in to the application. When the user logs out, the role is revoked.

> **See Also:**
>
> - *Oracle Database Security Guide* more information about privilege and role authorization
> - *Oracle Database Security Guide* more information about secure application roles

Example 5-1 shows a secure application role procedure that allows the user to log in during business hours (8 a.m. to 5 p.m.) from a specific set of work stations. If the user passes these checks, then the user is granted the `hr_admin` role and then is able to log in to the application.

Example 5-1 Secure Application Role Procedure to Restrict Access to Business Hours

```
CREATE OR REPLACE PROCEDURE hr_admin_role_check
 AUTHID CURRENT_USER
 AS
 BEGIN
  IF (SYS_CONTEXT ('userenv','ip_address')
    BETWEEN '192.0.2.10' and '192.0.2.20'
     AND
    TO_CHAR (SYSDATE, 'HH24') BETWEEN 8 AND 17)
  THEN
    EXECUTE IMMEDIATE 'SET ROLE hr_admin';
  END IF;
 END;
/
```

5.2 Automating Database Logins

To automate database logins, you create a logon trigger to run a PL/SQL procedure that can validate a user who is attempting to log in to an application. When the user logs in, the trigger executes. Logon triggers can perform multiple actions, such as generating an alert if the user fails the validation, displaying error messages, and so on.

Example 5-2 shows a simple logon trigger that executes a PL/SQL procedure.

Example 5-2 Creating a Logon Trigger

```
CREATE OR REPLACE TRIGGER run_logon_trig AFTER LOGON ON DATABASE
 BEGIN
  sec_mgr.check_user_proc;
 END;
```

> **See Also:**
>
> - *Oracle Database PL/SQL Language Reference* for detailed information about the `CREATE TRIGGER` statement
> - *Oracle Database Security Guide* for information about how to create a logon trigger that runs a database session application context package

5.3 Controlling User Access with Fine-Grained Access Control

There are several ways that you can control the level of access users have to data from your applications.

- **Oracle Virtual Private Database (VPD):** VPD enables you to create policies that can restrict database access at the row and column level. Essentially, VPD adds a dynamic `WHERE` clause to a SQL statement that is issued against the table, view, or synonym to which an VPD security policy was applied.

  For example, suppose the user enters the `SELECT * FROM OE.ORDERS` statement. A VPD policy can transform this statement to the following statement instead, so that only the sales representative who owns the order can view this data:

  ```
  SELECT * FROM OE.ORDERS
   WHERE SALES_REP_ID = SYS_CONTEXT('USERENV','SESSION_USER');
  ```

- **Oracle Data Redaction:** Oracle Data Redaction masks data at run time, at the moment the user attempts to access the data (that is, at query-execution time). This solution works well in a dynamic production system in which data is constantly changing. During the time that the data is being redacted, all data processing is performed normally, and the back-end referential integrity constraints are preserved. You typically redact sensitive data, such as credit card or Social Security numbers.

  You can mask the data in the following ways:

  - **Full redaction, in which the entire data is masked.** For example, the number 37828224 can be displayed as a zero.
  - **Partial redaction, in which only a portion of the data is redacted.** With this type, the number 37828224 can be displayed as *****224.
  - **Random redaction, in which the data is displayed as randomized data.** Here, the number 37828224 can appear as 93204857.
  - **Regular expressions, in which you redact data based on a search pattern.** You can use regular expressions in both full and partial redaction. For example, you can redact the user name of email addresses, so that only the domain shows: `jsmith` in the email address `jsmith@example.com` can be replaced with `[redacted]` so that the email address appears as `[redacted]@example.com`.
  - **No redaction, which enables you to test the internal operation of your redaction policies, with no effect on the results of queries against tables with policies defined on them.** You can use this option to test the redaction policy definitions before applying them to a production environment.

- **Oracle Label Security:** Oracle Label Security secures your database tables at the row level, and assigns these rows different levels of security based on the needs of your site. Rows that contain highly sensitive data can be assigned a label entitled `HIGHLY SENSITIVE`; rows that are less sensitive can be labeled as `SENSITIVE`, and so on. Rows that all users can have access to can be labeled `PUBLIC`. You can create as many labels as you need, to fit your site's security requirements.

  For example, when user `fred`, who is a low-level employee, logs in, he would see only data that is available to all users, as defined by the `PUBLIC` label. Yet when his director, `hortensia`, logs in, she can see all the sensitive data that has been assigned the `HIGHLY SENSITIVE` label.

- **Oracle Database Vault:** Oracle Database Vault enables you to restrict administrative access to your data. By default, administrators (such as user `SYS` with the `SYSDBA` privilege)

have access to all data in the database. Administrators typically must perform tasks such performance tuning, backup and recovery, and so on. However, they do not need access to your salary records. Database Vault enables you to create policies that restrict the administrator's actions yet not prevent them from performing normal administrative activities.

A typical Database Vault policy could, for example, prevent an administrator from accessing and modifying the `HR.EMPLOYEES` table. You can create fine-tuned policies that impose restrictions such as limiting the hours the administrators can log in, which computers they can use, whether they can log in to the database using dynamic tools, and so on. Furthermore, you can create policies that generate alerts if the administrator tries to violate a policy.

See Also:

- *Oracle Database Security Guide* for more information about Oracle Virtual Private Database
- *Oracle Database Advanced Security Guide* for more information about Oracle Data Redaction
- *Oracle Label Security Administrator's Guide* for more information about Oracle Label Security
- *Oracle Database Vault Administrator's Guide* for more information about Oracle Database Vault

5.4 Using Invoker's and Definer's Rights for Procedures and Functions

Topics:

5.4.1 What Are Invoker's Rights and Definer's Rights?

When you create a procedure or function (that is, a program unit), you can design it so that it runs with either the privileges of the owner (you) or the privileges of the person who is invoking it. Definer's rights run the program unit using the owner's privileges and invoker's rights run the program unit using the privileges of the person who runs it. For example, suppose user `harold` creates a procedure that updates the table `orders`. User `harold` then grants the `EXECUTE` privilege on this procedure to user `hazel`. If `harold` had created the procedure with definer's rights, then the procedure would expect the `orders` table to be in `harold`'s schema. If he created it with invoker's rights, then the procedure would look for the orders table in `hazel`'s schema.

To designate a program unit as definer's rights or invokers rights, use the `AUTHID` clause in the creation statement. If you omit the `AUTHID` clause, then the program unit is created with definer's rights.

Example 5-3 shows how to use the `AUTHID` clause in a `CREATE PROCEDURE` statement to specify definer's rights or invoker's rights.

Example 5-3 Creating Program Units with Definer's Rights or Invoker's Rights

```
CREATE PROCEDURE my_def_proc AUTHID DEFINER -- Definer's rights procedure
  AS ...

CREATE PROCEDURE my_inv_proc AUTHID CURRENT_USER -- Invoker's rights procedure
  AS ...
```

See Also:

- *Oracle Database PL/SQL Language Reference* for details about definer's rights and invoker's rights procedures and functions
- *Oracle Database PL/SQL Language Reference* for details about the `CREATE PROCEDURE` statement
- *Oracle Database PL/SQL Language Reference* for details about the `CREATE FUNCTION` statement

5.4.2 Protecting Users Who Run Invoker's Rights Procedures and Functions

An important consideration when you create an invoker's rights program unit is the level of privilege that the invoking users have. Suppose user `harold` is a low-ranking employee who has few privileges and `hazel` is an executive with many privileges. When `hazel` runs `harold`'s invoker's rights procedure, the procedure temporarily inherits `hazel`'s privileges (all of them). But because `harold` owns this procedure, he can modify it without her knowing it to behave in ways that take advantage of `hazel`'s privileges, such as giving `harold` a raise. To help safeguard against this type of scenario, after she has ensured that `harold` is trustworthy, user `hazel` can grant `harold` permission so that his invoker's rights procedures and functions have access to `hazel`'s privileges when she runs them. To do so, she must grant him the `INHERIT PRIVILEGES` privilege.

Example 5-4 shows how invoking user `hazel` can grant user `harold` the `INHERIT PRIVILEGES` privilege.

Example 5-4 Granting a Program Unit Creating the INHERIT PRIVILEGES Privilege

```
GRANT INHERIT PRIVILEGES ON hazel TO harold;
```

If `harold` proves untrustworthy, `hazel` can revoke the `INHERIT PRIVILEGES` privilege from him.

Administrators such as user `SYS` and `SYSTEM` have the `INHERIT ANY PRIVILEGES` privilege, which enable these users' invoker's rights procedures to have access to the privileges of any invoking users. As with all `ANY` privileges, grant this privilege only to trusted users.

See Also:

Oracle Database Security Guide for more information about managing security for definer's rights and invoker's rights procedures and functions

5.4.3 How Default Rights Are Handled for Java Stored Procedures

By default, Java class schema objects run with the privileges of their invoker, not with definer's rights. If you want your Java schema objects to run with definer's rights, then when you load them by using the `loadjava` tool, specify the `-definer` option.

Example 5-5 shows how to use the `-definer` option in a `loadjava` command.

Example 5-5 Loading a Java Class with Definer's Rights

```
loadjava -u joe -resolve -schema TEST -definer ServerObjects.jar
Password: password
```

You can use the `-definer` option on individual classes. Be aware that different definers may have different privileges. Apply the `-definer` option with care, so that you can achieve the desired results. Doing so helps to ensure that your classes run with no more than the privileges that they need.

> **See Also:**
>
> - *Oracle Database Java Developer's Guide* for detailed information about the `loadjava` tool
> - *Oracle Database Java Developer's Guide* for more information about controlling the current user in Java applications

5.5 Managing External Procedures for Your Applications

For security reasons, Oracle external procedures run in a process that is physically separate from Oracle Database. When you invoke an external procedure, Oracle Database creates the `extproc` operating system process (or agent), by using the operating system privileges of the user that started the listener for the database instance.

You can configure the `extproc` agent to run as a designated operating system credential. To use this functionality, you define a credential to be associated with the `extproc` process, which then can authenticate impersonate (that is, run on behalf of the supplied user credential) before loading a user-defined shared library and executing a function. To configure the `extproc` user credential, you use the PL/SQL package `DBMS_CREDENTIAL` and the PL/SQL statement `CREATE LIBRARY`.

> **See Also:**
>
> *Oracle Database Security Guide* for more information about securing external procedures

5.6 Auditing User Activity

You can create audit policies to audit specific actions in the database. Oracle Database then records these actions in an audit trail. The database mandatorily audits some actions and

writes these to the audit trail as well. The audit policies that you create can be simple, such as auditing all actions by a specific user, or complex, such as testing for specific conditions and sending email alerts if these conditions are violated.

When you install an Oracle Database, you can choose how your database is audited.

- **Unified auditing:** In unified auditing, all audit trails are written to a single audit trail, viewable by the `V$UNIFIED_AUDIT_TRAIL` and `GV$UNIFIED_AUDIT_TRAIL` dynamic views. This audit trail encompasses not only Oracle Database-specific actions, but also actions performed in Oracle Real Application Security, Oracle Recovery Manager, Oracle Database Vault, and Oracle Label Security environments. The audit records from these sources are presented in a consistent, uniform format. You can create named audit policies and enable and disable them as necessary. If you want to use unified auditing, then you must migrate your databases to it.
- **Mixture of unified auditing and pre-Release 12*c* auditing:** For the most part, this option enables you to use the pre-Release 12*c* auditing, in which audit records are written to different locations using their own formats. However, Release 12*c* functionality, such as using auditing in a multitenant environment, is available. This type of auditing is the default for both new and upgraded databases.

In both cases, when you upgrade your databases to Oracle Database 12*c* Release 1 (12.1.0.1), the audit records from the previous release are preserved. If you decide to migrate to use unified auditing fully, you can archive these earlier records and then purge the audit trail. After you complete the migration, the new audit records are written to the unified audit trail.

See Also:

- *Oracle Database Security Guide* for more information about creating and managing unified auditing policies
- *Oracle Database Security Guide* to find a detailed comparison of unified auditing and pre-Release 12*c* auditing
- *Oracle Database Upgrade Guide* for information about migrating to unified auditing