Oracle Database 12*c*: Security

Activity Guide – Volume I

D81599GC10

Edition 1.0

June 2014

D84091

Author

Dominique Jeunot

Technical Contributors and Reviewers

Jean-François Verrier, Donna Keesling, James Spiller, Gerlinde Frenzen, Pat Huey, Veerabhadra Rao Putrevu, Joel Goodman

This book was published using: Oracle Tutor

Table of Contents

Practices for Lesson 1: Introduction

Chapter 1

Practices for Lesson 1: Overview

Practices Overview

Background:

In the practices of this course, you assume the role of a database administrator (DBA) and of the security officer. The operating system (OS) accounts on your computer are:

- The `oracle` user with a password of `oracle`
- The `root` user with a password of `oracle`

Simple and easy-to-remember passwords will be used in order to not detract from the purpose of the exercise. In real development and production environments, use strong passwords following the guidelines presented in this course and in the *Oracle Database Security Guide 12c.*

The existing installation resides in the following `ORACLE_HOME`:

`/u01/app/oracle/product/12.1.0/dbhome_1`

You find the following instances and databases:

- The instance and non-container database, repository for Enterprise Manager Cloud Control: `em12rep`
- The instance and a non-container database: `orcl`
- The instance and multitenant container database: `cdb1`
- The pluggable database: `pdb1_1` within `cdb1`
- The pluggable database: `pdb1_2` within `cdb1`

The login information for the various connections is the following:

- Database accounts: `SYS` and `SYSTEM`, and all other new accounts and sample accounts are assigned the `oracle_4U` password.
- The security officer account is assigned a different password: `oracle_4sec`
- Enterprise Manager Cloud Control: `sysman` user is assigned the `Oracle123` password.

Practice 1-1: Environment Familiarization

Overview

In this practice, you will explore the server environment and create users for later practices.

Tasks

1. Verify that you are logged in as the `oracle` user when you right-click the desktop and click Open in Terminal to open a terminal window. The UID and GID may have different values than yours. Do not care about the values but do care about the user used to log in.

```
$ id
uid=54321(oracle) gid=54321(oinstall)
groups=54321(oinstall),54322(dba),54323(oper),54324(backupdba),5
4325(dgdba),54326(kmdba),54327(asmdba)
$
```

2. Before you start reviewing the practices environment, verify the permissions set for the labs scripts in `/home/oracle/labs` directory and the demos in `/home/oracle/labs/demos` directory. If the permissions are not appropriately set, execute the following UNIX commands to set the right permissions for all practices and demos. Then set the proper network aliases in the `tnsnames.ora` file.

```
$ su
Password: ******
# chown -R oracle:oinstall /home/oracle/labs
# exit
exit
$ chmod -R 777 /home/oracle/labs
$ cp /home/oracle/labs/admin/tnsnames.ora
/u01/app/oracle/product/12.1.0/dbhome_1/network/admin
$
```

3. Move the glogin.sql file to the appropriate destination. This script is automatically run each time you start SQL*Plus. This file avoids you having to enter SQL*Plus commands each time you disconnect and reconnect to SQL*Plus.

```
$ cp /home/oracle/labs/admin/glogin.sql
/u01/app/oracle/product/12.1.0/dbhome_1/sqlplus/admin
$
```

4. To list the running instances, you can search for the `SMON` background process. Any running instance includes the `SMON` background process at least.

```
$ pgrep -lf smon
1024 ora_smon_cdb1
4322 ora_smon_orcl
29667 ora_smon_em12rep
$
```

   There are three running instances, **`orcl`**, **`cdb1`** and **`em12rep`**. Notice that the user running the `orcl` and `cdb1` instances is **`oracle`**.

5. Connect to the `orcl` instance as the `SYS` user.

a. Use the `oraenv` utility to set the `ORACLE_SID` environment variable to the `orcl` value. The utility automatically sets the `ORACLE_HOME` to **`/u01/app/oracle/product/12.1.0/dbhome_1`**.

```
$ . oraenv
ORACLE_SID = [oracle] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

b. Use SQL*Plus to connect to the instance.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
SYS orcl >
```

c. View the instance name.

```
SYS orcl > select instance_name from v$instance;

INSTANCE_NAME
----------------
orcl

SYS orcl > exit
$
```

d. Execute the `create_users.sh` script to create new users in the database. The users `JIM`, `TOM` will be used in later practices. Make sure you are in the `~/labs/USERS` directory.

```
$ cd ~/labs/USERS
$ ./create_users.sh
$
```

6. Connect to the `cdb1` instance as the `SYS` user.

a. Use the `oraenv` utility to set the `ORACLE_SID` environment variable to the `cdb1` value.

```
$ . oraenv
ORACLE_SID = [orcl] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

b. Use SQL*Plus to connect to the instance.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
SYS cdb1 >
```

c. View the instance name.

```
SYS cdb1 > select instance_name from v$instance;

INSTANCE_NAME
----------------
cdb1

SYS cdb1 >
```

d. Quit the `SYS` session.

```
SYS cdb1 > EXIT
$
```

7. Connect to the `pdb1_1` pluggable database as the `SYS` user.

a. Use the `oraenv` utility to set the `ORACLE_SID` environment variable to the `cdb1` value. The instance for all pluggable databases in the `cdb1` container database is the `cdb1` instance.

```
$ . oraenv
ORACLE_SID = [cdb1] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

b. Use the service name of the PDB to connect to `pdb1_1`.

1) Explore the services to check if the `pdb1_1` service name is started.

```
$ lsnrctl status
…
Services Summary...
Service "em12rep" has 1 instance(s).
  Instance "em12rep", status READY, has 1 handler(s) for this
service...
Service "em12repxdb" has 1 instance(s).
  Instance "em12rep", status READY, has 1 handler(s) for this
service...
Service "cdb1" has 1 instance(s).
  Instance "cdb1", status READY, has 1 handler(s) for this
service...
```

```
Service "cdb1XDB" has 1 instance(s).
  Instance "cdb1", status READY, has 1 handler(s) for this
service...
Service "orcl" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
Service "orclXDB" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
Service "pdb1_1" has 1 instance(s).
  Instance "cdb1", status READY, has 1 handler(s) for this
service...
Service "pdb1_2" has 1 instance(s).
  Instance "cdb1", status READY, has 1 handler(s) for this
service...
The command completed successfully
$
```

2) Connect to `pdb1_1`.

```
$ sqlplus sys@pdb1_1 as sysdba

Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
SYS pdb1_1 >
```

3) View the instance name.

```
SYS pdb1_1 > select instance_name from v$instance;

INSTANCE_NAME
----------------
cdb1

SYS pdb1_1 >
```

4) View the pluggable database name.

```
SYS pdb1_1 > select name from v$pdbs;

NAME
----------------
PDB1_1

SYS pdb1_1 >
```

c. Connect to `pdb1_2`.

```
SQL pdb1_1 > CONNECT sys@pdb1_2 as sysdba
Enter password: ******
Connected.
SYS pdb1_2 >
```

1) View the instance name.

```
SYS pdb1_2 > select instance_name from v$instance;

INSTANCE_NAME
----------------
cdb1

SYS pdb1_2 >
```

2) View the pluggable database name.

```
SYS pdb1_2 > SHOW con_name

CON_NAME
------------------------------
PDB1_2
SYS pdb1_2 >
```

d. Quit the `SYS` session.

```
SYS pdb1_2 > EXIT
$
```

Practices for Lesson 2: Security Requirements

Chapter 2

Practices for Lesson 2: Overview

Practices Overview

The practices show how secure-coding practices can be used to reduce or eliminate the possibility of SQL injection exploits. The basic methods used in reducing the possibility of SQL injection can be adapted and applied to other common exploits. Specifics such as removing dynamic SQL would be changed to not allowing certain characters in XML or HTML to prevent cross-scripting. But general techniques such as peer review and testing are applicable across all type of exploits.

Practice 2-1: SQL Injection Exploit Tutorial *(optional)*

Overview

Without proper safeguards, applications are vulnerable to various forms of security attack. One particularly pervasive method of attack is called SQL injection. Using this method, a hacker can pass string input to an application with the hope of gaining unauthorized access to a database. By taking this self-study "*Defending Against SQL Injection Attacks!*" tutorial, you can arm yourself with techniques and tools to strengthen your code and applications against these attacks. This tutorial employs text and diagrams to present concepts, design issues, coding standards, processes, and tools.

Tasks

1. Launch a browser and enter: file:////home/oracle/labs/SQL_Injection/index.htm.

   Note: You may get an Adobe Flash Player 10 settings window when launching demos in the topics by clicking on the mouse image. This Adobe Flash Player 10 settings window includes the following messages:

   Adobe Flash Player has stopped a potentially unsafe operation

   The following local application on your computer or network:

   //home/oracle/labs/SQL_Injection/html/lesson1/les01_first_order_attack_skin.swf

   is trying to communicate with this Internet-enabled location:

   //home/oracle/labs/SQL_Injection/html/lesson1/les01_first_order_attack.htm

   To let this application communicate with the internet, click Settings.

   You must restart the application after changing your settings.

   Click the Settings button the first time you need to view a demo. Another window opens and let you view the demo. In this window, the lesson number and the file names vary based on the demos you launch for which topic.

   Note: If you click the link - *More ST Curriculum Tutorials*, it is not working displaying "504 Gateway Timeout". If you need more tutorials, go to Oracle Learning Library (OLL)

   Note: Please don't use the link Lesson 3.1 because the page navigates to 3.2 Use Static SQL instead of 3.1 Use Compile-Time-Fixed SQL Statement. If you really need to view this Lesson 3.1, enter file:////home/oracle/labs/SQL_Injection/html/lesson3/les03_tm_static1.htm in the browser.

Practice 2-2: Using Invoker's Rights Procedure

Overview

In this practice, you reduce SQL injection vulnerability by using invoker's rights.

If you do not provide an interface to an attacker, clearly it is not available to be abused. Thus, the first, and arguably the most important, line of your defense is to reduce the exposed interfaces to only those that are absolutely required. You can reduce the exposed interfaces by:

- Using invoker's rights to reduce SQL injection vulnerability
- Reducing arbitrary inputs

Stored program units and SQL methods execute with a set of privileges. By default, the privileges are those of a schema owner, also known as the definer. Definer's rights not only dictate the privileges, but are also used to resolve object references. If a program unit does not need to be executed with the escalated privileges of the definer, you should specify that the program unit executes with the privileges of a caller, also known as the invoker.

Tasks

1. Create a definer's rights procedure in the `orcl` instance. The `CHANGE_PASSWORD` procedure is created under the `SYS` schema. It accepts two parameters and uses them in the `ALTER USER` statement.

   a. Use the `oraenv` utility to set the `ORACLE_SID` environment variable to the `orcl` value.

```
$ . oraenv
ORACLE_SID = [cdb1] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

   b. Use SQL*Plus to connect to the instance.

```
$ sqlplus / as sysdba


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
SYS orcl >
```

   c. Create the `CHANGE_PASSWORD` procedure.

```
SYS orcl > CREATE OR REPLACE
PROCEDURE change_password(p_username VARCHAR2 DEFAULT NULL,
                          p_new_password VARCHAR2 DEFAULT NULL)
IS
  v_sql_stmt VARCHAR2(500);
BEGIN
  v_sql_stmt := 'ALTER USER '||p_username ||' IDENTIFIED BY '
                  || p_new_password;
```

```
  EXECUTE IMMEDIATE v_sql_stmt;
END change_password;
/
2    3    4    5    6    7    8    9   10   11

Procedure created.
SYS orcl >
```

Note the use of dynamic SQL with concatenated input values within the `v_sql_stmt` character string.

2. As the `SYS` user, grant `OE`, `HR`, and `SH` the ability to execute the `CHANGE_PASSWORD` procedure.

```
SYS orcl > GRANT EXECUTE ON change_password to OE, HR, SH;

Grant succeeded.

SYS orcl >
```

3. Result: Anyone that connects as `SH`, `OE`, or `HR` can change the password of any user, without knowing that user's password. Connect as `OE` to test that you can change the password of `SYS`.

```
SYS orcl > CONNECT oe
Enter password: ******
Connected.
OE orcl >
OE orcl > EXECUTE sys.change_password ('SYS', 'mine')

PL/SQL procedure successfully completed.

OE orcl >
```

4. Check that the password of `SYS` has changed.

```
OE orcl > CONNECT sys@orcl as sysdba
Enter password: ****** (oracle_4U)
ERROR:
ORA-01017: invalid username/password; logon denied

Warning: You are no longer connected to ORACLE.
    >
    > CONNECT sys@orcl as sysdba
Enter password: ****** (mine)
Connected.
SYS orcl >
```

5. Reset the `SYS` password to the initial value `oracle_4U`.

```
SYS orcl > EXECUTE sys.change_password ('SYS', 'oracle_4U')
```

```
PL/SQL procedure successfully completed.

SYS orcl >
```

6. To disallow another user from changing a password that does not belong to the user, redefine the `CHANGE_PASSWORD` procedure with the invoker's rights adding the `AUTHID CURRENT_USER` clause.

```
SYS orcl > CREATE OR REPLACE PROCEDURE change_password
      (p_username VARCHAR2 DEFAULT NULL,
       p_new_password VARCHAR2 DEFAULT NULL)
AUTHID CURRENT_USER
IS
  v_sql_stmt VARCHAR2(500);
BEGIN
  v_sql_stmt := 'ALTER USER '||p_username ||' IDENTIFIED BY '
                 || p_new_password;
  EXECUTE IMMEDIATE v_sql_stmt;
END change_password;
/
2    3    4    5    6    7    8    9   10   11   12
Procedure created.

SYS orcl >
```

7. Reconnect as `OE` to test that you cannot change the password of `SYS`.

```
SYS orcl > CONNECT oe
Enter password: ******
Connected.
OE orcl > EXECUTE sys.change_password ('SYS', 'yours')
BEGIN sys.change_password('SYS', 'yours'); END;

*
ERROR at line 1:
ORA-01031: Insufficient privileges
ORA-06512: at "SYS.CHANGE_PASSWORD", at line 10
ORA-06512: at line 1

OE orcl >
```

8. Use the same procedure to change OE password.

```
OE orcl > EXECUTE sys.change_password ('OE', 'oracle')

PL/SQL procedure successfully completed.
```

```
OE orcl > EXECUTE sys.change_password ('OE', 'oracle_4U')

PL/SQL procedure successfully completed.

OE orcl >
```

.

Practice 2-3: Using Static SQL and Bind Arguments

Overview

Poor application design can lead to “designed in” vulnerabilities, where there are no apparent coding problems and everything works as intended.

However, you must design your code such that it is (ideally) entirely free of SQL injection vulnerabilities, or contains measures that mitigate the impact of a successful attack.

The common flaw of all code vulnerable to SQL injection is the construction of dynamic SQL by using string concatenation. Complete immunity from SQL injection attack can be achieved only through the elimination of input string concatenation in dynamic SQL.

- Avoid input string concatenation.
- Use bind arguments, whether automatically via static SQL or explicitly via dynamic SQL statements. Bind arguments are immune to SQL injection.

Design your code to use bind arguments wherever possible. The only exceptions should be when you need to concatenate identifiers or keywords because you have no other choice.

In this practice, you will create a SQL code to demonstrate SQL injection in `LIKE` operators and how to redefine the code to prevent SQL injection.

Tasks

1. Define two `LIST_PRODUCTS` procedures. The `LIST_PRODUCTS_DYNAMIC` procedure does not use bind arguments but contains concatenated input values. The `LIST_PRODUCTS_STATIC` procedure uses bind arguments.
   Create the `LIST_PRODUCTS_DYNAMIC` procedure containing dynamic SQL with concatenated input values. Why is the SQL considered as dynamic? The ‘*SELECT product_name, min_price, list_price FROM products WHERE product_name like ‘’%’||p_product_name||’%’’*” statement is unresolved at compile-time.

```
OE orcl > CONNECT oe
Enter password: ******
Connected.
OE orcl > SET SERVEROUTPUT ON
OE orcl > CREATE OR REPLACE PROCEDURE list_products_dynamic
           (p_product_name VARCHAR2 DEFAULT NULL)
AS
   TYPE cv_prodtyp IS REF CURSOR;
   cv   cv_prodtyp;
   v_prodname products.product_name%TYPE;
   v_minprice products.min_price%TYPE;
   v_listprice products.list_price%TYPE;
   v_stmt  VARCHAR2(400);
BEGIN
v_stmt := 'SELECT product_name, min_price, list_price
           FROM   products
           WHERE  product_name like ''%'||p_product_name||'%''';
OPEN cv FOR v_stmt;
dbms_output.put_line(v_stmt);
```

```
  LOOP
      FETCH cv INTO v_prodname, v_minprice, v_listprice;
      EXIT WHEN cv%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Product Info: '||v_prodname ||', '||
                           v_minprice ||', '|| v_listprice);
  END LOOP;
  CLOSE cv;
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16   17   18   19   20   21   22   23   24
Procedure created.

OE orcl >
```

2. Execute the procedure.

```
OE orcl > EXEC list_products_dynamic('Laptop')
SELECT product_name, min_price, list_price
       FROM   products

WHERE  product_name like '%Laptop%'
Product Info: Laptop 128/12/56/v90/110, 2606, 3219
Product Info: Laptop 16/8/110, 800, 999
Product Info: Laptop 32/10/56, 1542, 1749
Product Info: Laptop 48/10/56/110, 2073, 2556
Product Info: Laptop 64/10/56/220, 2275, 2768

PL/SQL procedure successfully completed.

OE orcl >
```

The result is correct because the user entered an appropriate product name.

3. Execute the procedure performing a SQL injection attack and see that you can retrieve the list of database accounts.

```
OE orcl > EXEC list_products_dynamic(''' and 1=0 union select
cast(username as nvarchar2(100)), null, null from all_users --')
SELECT product_name, min_price, list_price
       FROM   products
WHERE  product_name like '%' and 1=0 union select cast(username
as
nvarchar2(100)), null, null from all_users --%'
Product Info: ANONYMOUS, ,
Product Info: APEX_040200, ,
Product Info: APEX_PUBLIC_USER, ,
Product Info: APPQOSSYS, ,
```

```
Product Info: AUDSYS, ,
Product Info: BI, ,
Product Info: CTXSYS, ,
Product Info: DBSNMP, ,
Product Info: DIP, ,
Product Info: DVF, ,
Product Info: DVSYS, ,
Product Info: FLOWS_FILES, ,
Product Info: GSMADMIN_INTERNAL, ,
Product Info: GSMCATUSER, ,
Product Info: GSMUSER, ,
Product Info: HR, ,
Product Info: IX, ,
Product Info: JIM, ,
Product Info: LBACSYS, ,
Product Info: MDDATA, ,
Product Info: MDSYS, ,
Product Info: OE, ,
Product Info: OJVMSYS, ,
Product Info: OLAPSYS, ,
Product Info: ORACLE_OCM, ,
Product Info: ORDDATA, ,
Product Info: ORDPLUGINS, ,
Product Info: ORDSYS, ,
Product Info: OUTLN, ,
Product Info: PM, ,
Product Info: SCOTT, ,
Product Info: SH, ,
Product Info: SI_INFORMTN_SCHEMA, ,
Product Info: SPATIAL_CSW_ADMIN_USR, ,
Product Info: SPATIAL_WFS_ADMIN_USR, ,
Product Info: SYS, ,
Product Info: SYSBACKUP, ,
Product Info: SYSDG, ,
Product Info: SYSKM, ,
Product Info: SYSTEM, ,
Product Info: TOM, ,
Product Info: WMSYS, ,
Product Info: XDB, ,
Product Info: XS$NULL, ,

PL/SQL procedure successfully completed.
```

```
OE orcl >
```

Notice the SQL injection attack succeeded through the concatenation of a `UNION` set operator to the dynamic SQL statement.

4. Create the `LIST_PRODUCTS_STATIC` procedure that contains static SQL with bind arguments. Why is SQL considered as static? Although the '*SELECT product_name, min_price, list_price FROM products WHERE product_name like v_bind*' statement is not a compile-time-fixed SQL statement text, the SQL syntax, however, is frozen at compile time. It is clear that the SQL statement extracts the prices of the `product_name` specified by the bind variable `v_bind`. This kind of statement is a run-time static SQL statement.

```
OE orcl > CREATE OR REPLACE PROCEDURE list_products_static
               (p_product_name VARCHAR2 DEFAULT NULL)
AS
  v_bind  VARCHAR2(400);
BEGIN
  v_bind := '%'||p_product_name||'%';
  FOR i in
  (SELECT product_name, min_price, list_price
   FROM   products
   WHERE  product_name like v_bind)
  LOOP
    DBMS_OUTPUT.PUT_LINE('Product Info: '||i.product_name
||','||
                           i.min_price ||', '|| i.list_price);
  END LOOP;
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16
Procedure created.

OE orcl >
```

5. Execute the procedure.

```
OE orcl > EXEC list_products_static('Laptop')
Product Info: Laptop 128/12/56/v90/110,2606, 3219
Product Info: Laptop 16/8/110,800, 999
Product Info: Laptop 32/10/56,1542, 1749
Product Info: Laptop 48/10/56/110,2073, 2556
Product Info: Laptop 64/10/56/220,2275, 2768

PL/SQL procedure successfully completed.
OE orcl >
```

Notice that the procedure runs correctly with a "normal" input.

6. Execute the same static procedure to verify that it is not vulnerable to SQL injection.

```
OE orcl > EXEC list_products_static(''' and 1=0 union select
cast(username as nvarchar2(100)), null, null from all_users --')

PL/SQL procedure successfully completed.

OE orcl >
```

Notice that the SQL injection attempt failed.

Practice 2-4: Avoiding SQL Injection Through Dynamic PL/SQL block

Overview

In this practice, you will create a code to demonstrate SQL injection through dynamic PL/SQL block and redefine the code to prevent SQL injection. The practice shows how SQL injection via dynamic PL/SQL can be even more dangerous than via dynamic SQL.

Tasks

1. Create the `GET_AVG_SALARY` function containing a dynamic PL/SQL block used to retrieve the average salary with a concatenated input parameter `p_job`. This is a SQL injection vulnerability.

```
OE orcl > CONNECT hr
Enter password: ******
Connected.
HR orcl > SET SERVEROUTPUT ON
HR orcl > CREATE OR REPLACE FUNCTION get_avg_salary (p_job
VARCHAR2)
RETURN NUMBER
AS
  avgsal employees.salary%TYPE;
  v_blk VARCHAR2(4000);
BEGIN
  v_blk := 'BEGIN
             SELECT AVG(salary) INTO :avgsal
             FROM   hr.employees
             WHERE  job_id = '''||P_JOB||'''; END;
            ';
  EXECUTE IMMEDIATE v_blk
  USING OUT avgsal;
  dbms_output.put_line('Code: ' || v_blk);
  RETURN avgsal;
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16   17
Function created.
HR orcl >
```

2. Execute the dynamic PL/SQL block.

```
HR orcl > exec dbms_output.put_line('Average salary is: ' ||
get_avg_salary('SH_CLERK'))

Code: BEGIN
         SELECT AVG(salary) INTO :avgsal
         FROM
```

```
hr.employees
         WHERE  job_id = 'SH_CLERK'; END;

Average salary is: 3215

PL/SQL procedure successfully completed.

HR orcl >
```

It works fine and provides the correct result.

3. You will now attempt to change the salary of an employee although the function exists to show the average of a salary for a job.

```
HR orcl > select salary from employees where email='PFAY';

    SALARY
----------
      6000

HR orcl > exec dbms_output.put_line('Average salary is: ' ||
get_avg_salary('SH_CLERK''; UPDATE hr.employees SET salary=4500
WHERE email=''PFAY''; COMMIT; END;--'))

Code: BEGIN
         SELECT AVG(salary) INTO :avgsal
         FROM
hr.employees
         WHERE  job_id = 'SH_CLERK'; UPDATE hr.employees SET
salary=4500 WHERE email='PFAY'; COMMIT; END;--'; END;

Average salary is: 3215

PL/SQL procedure successfully completed.

HR orcl >
```

The `UPDATE` statement was injected successfully.

4. Check the salary of the `PFAY` employee.

```
HR orcl > select salary from employees where email='PFAY';

    SALARY
----------
      4500
HR orcl >
```

The salary updated has also been committed. Multiple statements can be injected through a PL/SQL block.

5. Reset the salary of the `PFAY` employee to 6000.

```
HR orcl > UPDATE hr.employees SET salary=6000 WHERE
email='PFAY';


1 row updated.


HR orcl > COMMIT;


Commit complete.


HR orcl >
```

6. Redefine the function so as to eliminate the SQL injection vulnerability by using an IN bind argument, `p_job`, with the dynamic PL/SQL.

```
HR orcl > CREATE OR REPLACE FUNCTION get_avg_salary (p_job
VARCHAR2)
RETURN NUMBER
AS
  avgsal employees.salary%TYPE;
  v_blk VARCHAR2(4000);
BEGIN
  v_blk := 'BEGIN
             SELECT AVG(salary) INTO :avgsal
             FROM   hr.employees
             WHERE  job_id = :p_job; END;
            ';
  EXECUTE IMMEDIATE v_blk
  USING OUT avgsal, IN p_job;
  dbms_output.put_line('Code: ' || v_blk);
  RETURN avgsal;
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16   17
Function created.


HR orcl >
```

7. Retest the new function and verify that the new code still works for a valid input.

```
HR orcl > exec dbms_output.put_line('Average salary is: ' ||
get_avg_salary('SH_CLERK'))


Code: BEGIN
```

```
          SELECT AVG(salary) INTO :avgsal
          FROM
hr.employees
          WHERE  job_id = :p_job; END;

Average salary is: 3215

PL/SQL procedure successfully completed.

HR orcl >
```

8. Retest the new function and verify that the new code does not work for an invalid input with the same SQL injection attack.

```
HR orcl > select salary from employees where email='PFAY';

    SALARY
----------
      6000

HR orcl > exec dbms_output.put_line('Average salary is: ' ||
get_avg_salary('SH_CLERK''; UPDATE hr.employees SET salary=4500
WHERE email=''PFAY''; COMMIT; END;--'))

Code: BEGIN
          SELECT AVG(salary) INTO :avgsal
          FROM
hr.employees
          WHERE  job_id = :p_job; END;

Average salary is:

PL/SQL procedure successfully completed.

HR orcl >
```

The block executes but returns a NULL value for the average salary because no `JOB_ID` column value matched the `'SH_CLERK''; UPDATE hr.employees SET salary=4500 WHERE email=''PFAY''; COMMIT; END;--'` value.

9. Check the salary of the `PFAY` employee.

```
HR orcl > select salary from employees where email='PFAY';

    SALARY
----------
      6000
```

```
HR orcl >
```

The `UPDATE` statement was not executed. The SQL injection failed.

Practice 2-5: Validating Input Using the DBMS_ASSERT Package

Overview

To guard against SQL injection in applications that do not use bind arguments with dynamic SQL, you must filter and sanitize concatenated strings. The primary use case for dynamic SQL with string concatenation is when an Oracle identifier (such as a table name or a user name) is unknown at code compilation time.

`DBMS_ASSERT` is an Oracle-supplied PL/SQL package containing functions that can be used to filter and sanitize input strings, particularly those that are meant to be used as Oracle identifiers.

In this practice, you will improve the `CHANGE_PASSWORD` procedure avoiding inappropriate input values for the user name and the password. Using bind arguments like in the previous practice is not possible in a DDL statement.

Use several `DBMS_ASSERT` functions to filter and sanitize the input values:

- `ENQUOTE_NAME` function to enclose the user's name in double quotes.
- `SCHEMA_NAME` function to verify that the input string is an existing user name.
- `SIMPLE_SQL_NAME` function to verify that the password is a simple SQL name. The input value must meet the following conditions:
  - The name must begin with an alphabetic character. It may contain alphanumeric characters as well as the characters _, $, and # in the second and subsequent character positions.
  - Quoted SQL names are also allowed.
  - Quoted names must be enclosed in double quotes.
  - Quoted names allow any characters between the quotes.
  - Quotes inside the name are represented by two quote characters in a row, for example, "a name with "" inside" is a valid quoted name.
  - The input parameter may have any number of leading and/or trailing white space characters.
  - The length of the name is not checked.

Tasks

1. The input for the user name is user-supplied and so is in a normal identifier format. It needs to be pre-processed using a conversion routine. Create a function that converts a normal quoted value to an internal format value when a user-supplied value is to be used as a bind argument for a lookup of an internal object name.

```
HR orcl > CONNECT / AS SYSDBA
Connected.
SYS orcl > CREATE OR REPLACE FUNCTION toInternal(Id varchar2)
RETURN varchar2 IS
  Temp varchar2(40);
begin
  Temp := trim(Id);
-- See comments in text re trimming
-- Remove quotes
  IF substr(Temp,1,1) = '"' AND
```

```
     substr(Temp,length(Temp),1) = '"' then
     Temp := substr(Temp,2,length(Temp)-2);
  else
-- Not quoted, so make sure is upper case
     Temp := nls_upper(Temp);
  end IF;
RETURN Temp;
end;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16   17

Function created.

SYS orcl >
```

2. Redefine the `CHANGE_PASSWORD` procedure by using `DBMS_ASSERT` checking functions.

```
SYS orcl > SET SERVEROUTPUT ON
SYS orcl > CREATE OR REPLACE PROCEDURE change_password
               (p_username IN VARCHAR2,
                p_password IN VARCHAR2)
AUTHID CURRENT_USER
AS
v_stmt VARCHAR2(4000);
BEGIN
 v_stmt :=
   'ALTER USER '|| sys.dbms_assert.enquote_name(
sys.dbms_assert.schema_name(toInternal(p_username)),FALSE) ||
               ' IDENTIFIED BY '
     || sys.dbms_assert.simple_sql_name(p_password);
 DBMS_Output.Put_Line('SQL stmt: '|| v_stmt);
 EXECUTE IMMEDIATE v_stmt;
EXCEPTION WHEN OTHERS THEN
   RAISE;
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16   17   18
Procedure created.

SYS orcl >
```

3. Check that the procedure does not allow any invalid input value for the password.

```
SYS orcl > CONNECT hr
```

```
Enter password: ******
Connected.
HR orcl > SELECT default_tablespace from user_users
     WHERE  username='HR';
  2
DEFAULT_TABLESPACE
------------------------------
USERS

HR orcl > EXEC sys.change_password('hr','hr default tablespace
system quota unlimited on system')
BEGIN sys.change_password('hr','hr default tablespace system
quota unlimited on system'); END;

*
ERROR at line 1:
ORA-44003: invalid SQL name
ORA-06512: at "SYS.CHANGE_PASSWORD", line 16
ORA-06512: at line 1

HR orcl > SELECT default_tablespace from user_users
     WHERE  username='HR';
  2
DEFAULT_TABLESPACE
------------------------------
USERS

HR orcl >
```

4. Check that the procedure does not allow any invalid input value for the user name.

```
HR orcl > EXEC sys.change_password('hr oe','hr')
BEGIN sys.change_password('hr oe','hr'); END;

*
ERROR at line 1:
ORA-44001: invalid schema
ORA-06512: at "SYS.CHANGE_PASSWORD", line 16
ORA-06512: at line 1

HR orcl >
```

```
HR orcl > CONNECT hr
Enter password: ******
Connected.
HR orcl > EXIT
```

```
$
```

Practices for Lesson 3: Security Solutions

Chapter 3

Practices for Lesson 3: Overview

Practices Overview

The first practice is a case study where for each of the following scenarios, you can suggest security solutions. There is more than one correct solution for each scenario.

The other practices ask you to set up miscellaneous solutions appropriate to the situation.

Practice 3-1: Choosing Oracle Solutions

Overview

In this practice, you suggest security solutions according to each scenario.

Scenario 1

Your company sends backup tapes off site to a disaster recovery site. Payment information (including credit card numbers, customer names, and addresses) is in the data files included on the tapes. The PCI_DSS requirement 3 says “Protect stored cardholder data” and requirement 4 says “Encrypt transmission of cardholder data across open, public networks.” The chief information officer (CIO) wants to secure this information to prevent bad publicity if the backup tapes are lost or stolen, or if any cardholder information is acquired by intercepting network traffic.

Answer

Oracle Net Services enables you to use native network encryption for all Oracle Network traffic. Oracle Advanced Security allows you to use Transparent Data Encryption (TDE); the sensitive data in the database files will be encrypted. Thus, the image file backups will contain encrypted data. Using RMAN with Oracle Secure Backup to tape will ensure that the tape backup files are encrypted. Using RMAN can allow you to ensure that sensitive data is encrypted on backup sets to disk.

Scenario 2

The network security officer has detected abnormal activity involving port 1521 through a firewall and several desktop machines inside the firewall. The normal activity is for users outside the firewall to contact an application server; therefore, all the database activity should be through the application server and not on port 1521 through the firewall.

Answer

Port 1521 is the default port for the Oracle database listener. This may indicate an attempt to attack the database. Some or all of the following protections can be implemented.

- Port 1521 should be closed through the firewall. The only outside users allowed through the firewall contact the application server on its listener port (usually, this is an HTTP or HTTPS port, not port 1521).
- The database can be configured to accept connections only from the application server and to reject connections from any other machine.
- A good practice is to place the application server in one zone and the database in another zone with a firewall between them.

Scenario 3

The company is considering outsourcing the DBA activities to a third party. The concern is that a DBA who is not an employee will be able to access company-proprietary information, customer financial information, and employee medical information.

Answer

There are powerful system privileges assigned to the DBA role that allow the DBA to view data. There are two main solutions:

- Oracle Database Vault can be very easily configured to limit the data that the DBA can view.

- Use application-based encryption to encrypt sensitive data. Use a scheme that does not allow the DBA to access the encryption keys stored in a keystore. Use the `SYSKM` administrative privilege to hand over the key management to someone else.

Scenario 4

The current DBA has been granted the `SYSDBA` role to effectively start up and shut down the database instance, and use RMAN to make database backups. There have been some incidents in the past when company confidential information has been discovered on the Web. How can the current DBA protect himself or herself from accusations that he or she is the most likely suspect for any further security breaches because he or she had access?

Answer

There are two situations:

- The DBA has not yet migrated to unified auditing:
  - He can enable the `AUDIT_SYS_OPERATIONS` parameter to record every command that the `SYS` user issues.
  - In addition, the DBA can send these records to the `SYSLOG` facility, setting the `AUDIT_SYSLOG_LEVEL` parameter, so that the records can be written to an OS account to which he or she has no access.
  - In a CDB, the scope of the settings for this initialization parameter is the CDB. Although the audit trail is provided per PDB in a CDB, this initialization parameter cannot be configured for individual PDBs.
- The DBA has migrated to unified auditing, there is no action to take:
  - The `AUDIT_SYS_OPERATIONS` and `AUDIT_SYSLOG_LEVEL` parameters have no effect anymore.
  - When `SYS` is connected as `SYSDBA`, `SYSOPER`, `SYSBACKUP`, `SYSASM`, `SYSKM`, or `SYSDG`, it is subjected to all top-level statements, such as `STARTUP`, `SHUTDOWN`, `ALTER DATABASE`, and `ALTER SYSTEM`, until the database opens.
  - Unlike other Oracle Database components, Oracle RMAN events such as `BACKUP`, `RESTORE` and `RECOVER` are systematically audited and this is not necessary to create and enable an audit policy.

A complementary solution is to use Oracle Audit Vault to centralize all the audit records and keep them in a safe repository.

Practice 3-2: Configuring Monitoring Credentials Using Enterprise Manager Cloud Control

Overview

In this practice, you act as an Enterprise Manager administrator. You access Oracle Enterprise Manager Cloud Control 12*c* as the `sysman` user with the `Oracle123` password. You create the `credorcl` credential used for any connection as `SYS` user sharable in the database instance `orcl`.

Tasks

1. You check that the Enterprise Manager Cloud Control is available. Click the Firefox icon on the top panel (toolbar region) above the desktop to open a browser to access the Enterprise Manager Cloud Control console.
2. Enter the URL for Cloud Control: `https://<em_server_hostname>.<domain>:7802/em`. In the current setup, use `https://localhost:7802/em`. If an error appears, you must first start the OMS, else proceed directly with step 3.
   a. Start the Enterprise Manager Repository Database `em12rep` if not started already.

```
$ . oraenv
ORACLE_SID = [orcl] ? em12rep
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to an idle instance.
SYS em12rep > startup
ORACLE instance started.

Total System Global Area  400846848 bytes
Fixed Size                  2271568 bytes
Variable Size             339740336 bytes
Database Buffers           50331648 bytes
Redo Buffers                8503296 bytes
Database mounted.
Database opened.
SYS em12rep > EXIT
$
```

   b. Restart the OMS.

```
$ export OMS_HOME=/u01/app/oracle/product/middleware/oms
$ $OMS_HOME/bin/emctl start oms
Oracle Enterprise Manager Cloud Control 12c Release 2
Copyright (c) 1996, 2012 Oracle Corporation.  All rights
reserved.
```

```
Starting Oracle Management Server...
Starting WebTier...
WebTier Successfully Started
Oracle Management Server Successfully Started
Oracle Management Server is Up
WARNING: Limit of open file descriptors is found to be 1024.
The OMS has been started but it may run out of descriptors under
heavy usage.
For proper functioning of OMS, please set "ulimit -n" to be at
least 4096.
$
```

3. Most likely, you receive an “Untrusted Connection’ message and you need to add a security exception.
   a. At the end of the alert box, click **I Understand the Risks**.
   b. At the bottom of the page, click **Add Exception**.
   c. Confirm that “Permanently store this exception” is selected in your training environment and click **Confirm Security Exception**.
4. The Enterprise Manager Cloud Control console appears.
5. Enter **`sysman`** in the User Name field and **`Oracle123`** in the Password field. Then click **Login**.
6. The first time a new user logs in to Enterprise Manager, a page asks you to accept the license agreement. You have to accept only once. Then each time you will log in to Enterprise Manager, you will not get the license agreement page.

7. Then the “Select Enterprise Manager Home” page appears with choices, such as:
   - Summary
   - Databases
   - Incidents
   - SOA

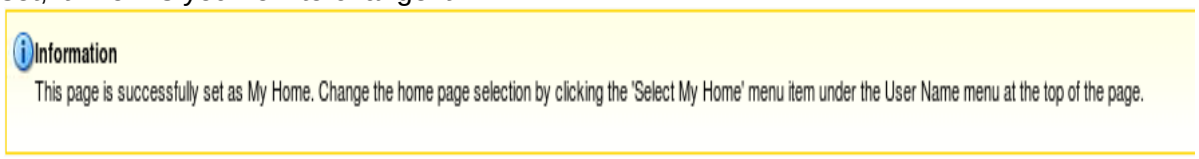
- Middleware
- Composite Application
- Service Request
- Services
- Business Applications
- Compliance Dashboard

Each choice has a Preview and a Select As My Home button.

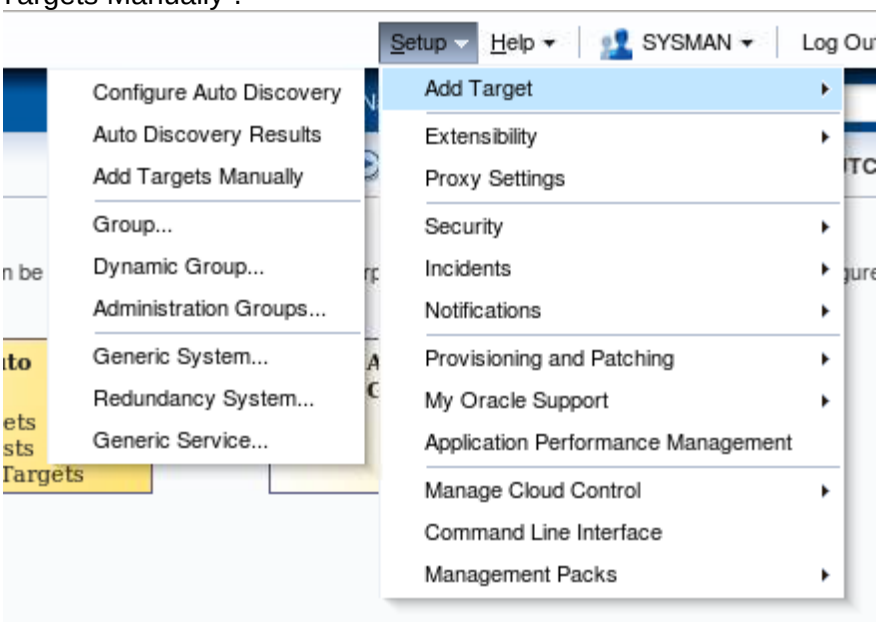
The page also has global menus with the following choices: Enterprise, Targets, Favorites, History, and Search Target Name (next to the search entry field). Each of the menu items has drop-down menus with further choices.

Preview any images that interest you.

8. Click “Select As My Home” in the top right hand corner of the page. After being successfully set, it informs you how to change it.

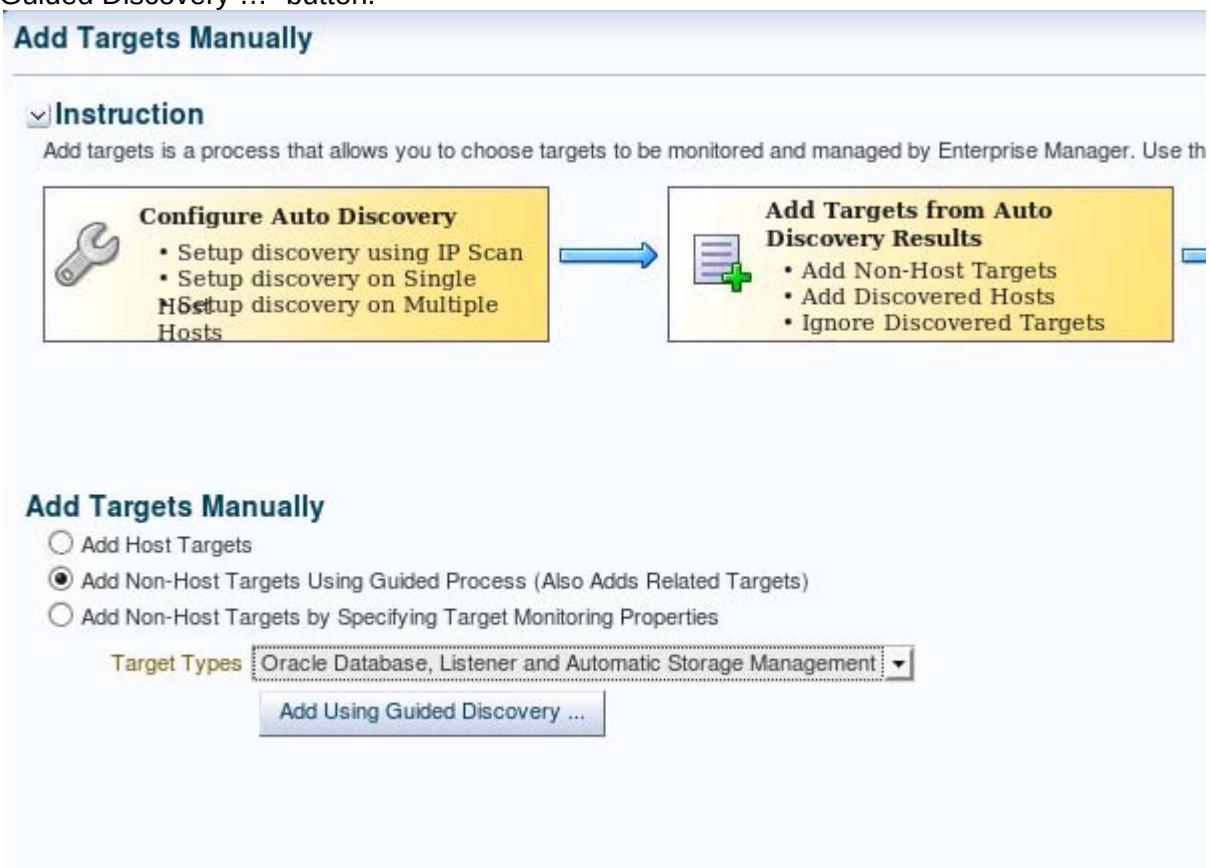


9. Add the `orcl` Database Instance as a new target in Enterprise Manager Cloud Control.
   a. In the top right hand corner of the page, click the “Setup” > “Add Target” > “Add Targets Manually”.



   b. In “Add Targets Manually”, choose “Add Non-Host Targets Using Guided Process (Also Adds Related Targets)”. Then in “Target Types”, choose “Oracle Database, Listener and Automatic Storage Management” for “Target Type”. Click “Add Using

Guided Discovery …" button.



c. In "Add Database Instance target: Specify Host", click the magnifying glass to find your host. Select your host, then click "Continue".

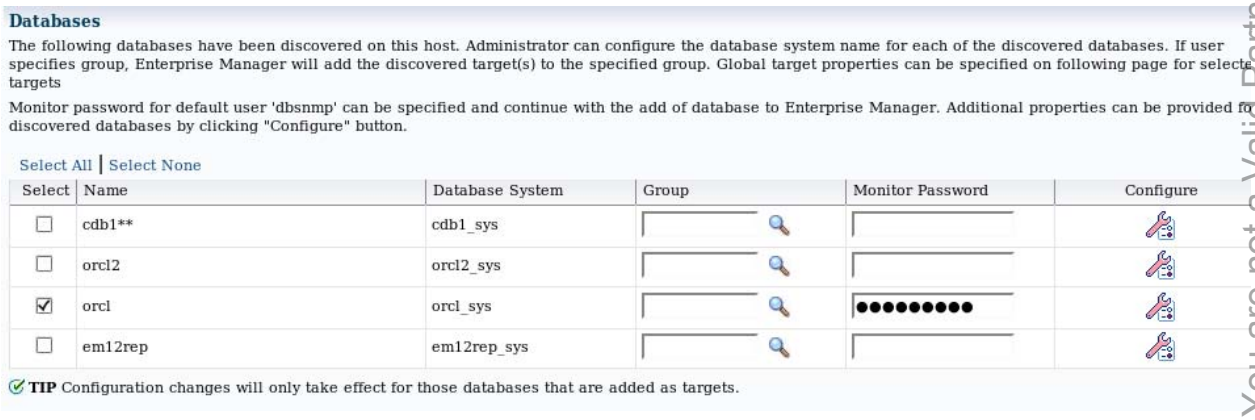
d. In the "Databases" list, deselect all databases except `orcl`. Deselect the listener.

1) Unlock the `DBSNMP` user. This user is the monitoring user used to test the connection once the target is being added. Open a terminal window.

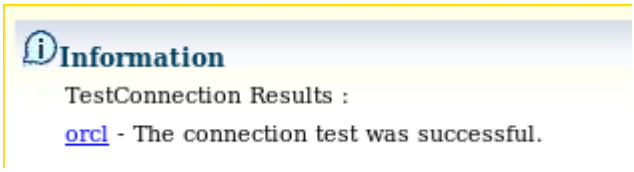
```
$ . oraenv
ORACLE_SID = [em12rep] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
```

```
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
SYS orcl > alter user dbsnmp identified by oracle_4U account
unlock;

User altered.

SYS orcl > EXIT
$
```

2) Enter `oracle_4U` for the "Monitor Password".



e. Click the "Test Connection" button. You should receive the following message:



f. Click the "Finish" then "Save" buttons to complete the operation, and finally "OK".

10. To create the monitoring credentials for your `orcl` database credentials, navigate to Setup > Security > Named Credentials. Click **Create**.

a. Enter the following values, then complete the **Access Control** section:

Field	Choice or Value
General Properties	
Credential Name	**`credorcl`**
Credential description	**`Credentials for Database`**
Authenticating Target Type	**`Database Instance`**
Credential type	**`Database Credentials`**
Scope	**`Target`**
Target type	**`Database Instance`**
Target Name	**`orcl (Click the magnifying glass`**

Field	Choice or Value
	`to find orcl and select)`
Credential Properties	
Username	**`SYSTEM`**
Password	**`oracle_4U`**
Confirm Password	**`oracle_4U`**
Role	**`NORMAL`**

b. Specify who can share, edit or even delete this shared credential by using one of the three privileges (`Full`, `Edit`, `View`).
   - `SYS` user with Full privilege will be able to use, edit, and delete the credential.
   - `SYSTEM` user with Edit privilege will be able to use and edit the credential.

   1) Click “Add Grant” then select the user `SYS` to be added in the Access Control list.
   2) Repeat this operation to add the user `SYSTEM`.
      By default, the selected users are granted the View privilege only.
   3) To grant Full privilege to `SYS`, select the `SYS` user and click “Change Privilege”. Choose Full and click OK.
   4) To grant Edit privilege to `SYSTEM`, select the `SYSTEM` user and click “Change Privilege”. Choose Edit and click OK.

11. Test against the `orcl` database instance, click **Test and Save** until you get the following message: **Confirmation Credential Operation Successful**. This means that the credential was successful and saved.
12. Test the `credorcl` named credential to connect to `orcl` database.
    a. Click **Targets** and then select **Databases**.
    b. Click the “Search List” radio button.
    c. Click the `orcl` link.
    d. Click **Administration**, then **Security** and then **Users.** The named credential `credorcl` is displayed.
    e. Click **Login** if you accept this named credential to log in the `orcl` database else choose **New** to define new login username and password.

Practice 3-3: Viewing Compliance Frameworks

Overview

In this practice, you will view the PCI DSS (Version 2) compliance framework and the Oracle Generic Compliance Framework supplied in Enterprise Manager Cloud Control. With the compliance feature, the former policies and groups of policies in Enterprise Manager Grid Control have been reinvented in a new hierarchy defined in Enterprise Manager Cloud Control 12c. This starts off with rules that are specific items to check for a particular target type.
What are the rules: they are checks, tests performed against the environment, for example — Is a parameter value set properly as per best practice guidelines?

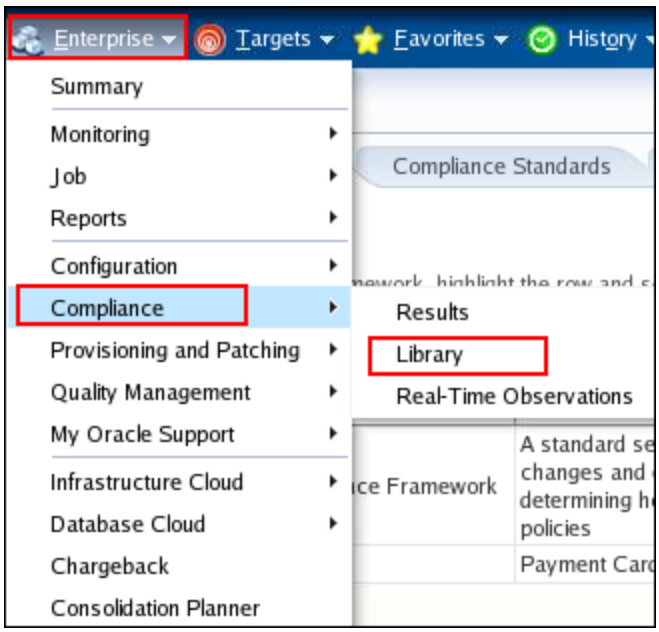
There are three kinds of rules:

- The first one: the repository rule which is very similar to the user-defined policy that we had in Enterprise Manager 11g. A repository rule is evaluated against the repository data only when the data changes underneath, but it uses the current data that exists in the repository. And we provide a repository browser to aid in rule creation to build the query.
- The second type of rule is the real-time rule that activates the agent to perform real-time change detection for file actions, for schema actions, process actions to detect when, where a particular action took place and who performed the action. And again you can apply the rule to a particular target type. This also detects unauthorized changes and correlate them to the Change Management System.
- The third type of rule is the Weblogic rule that performs BEA Guardian health checks integrated in Enterprise Manager. You can apply out-of-the-box 1300 rules to a particular target type.

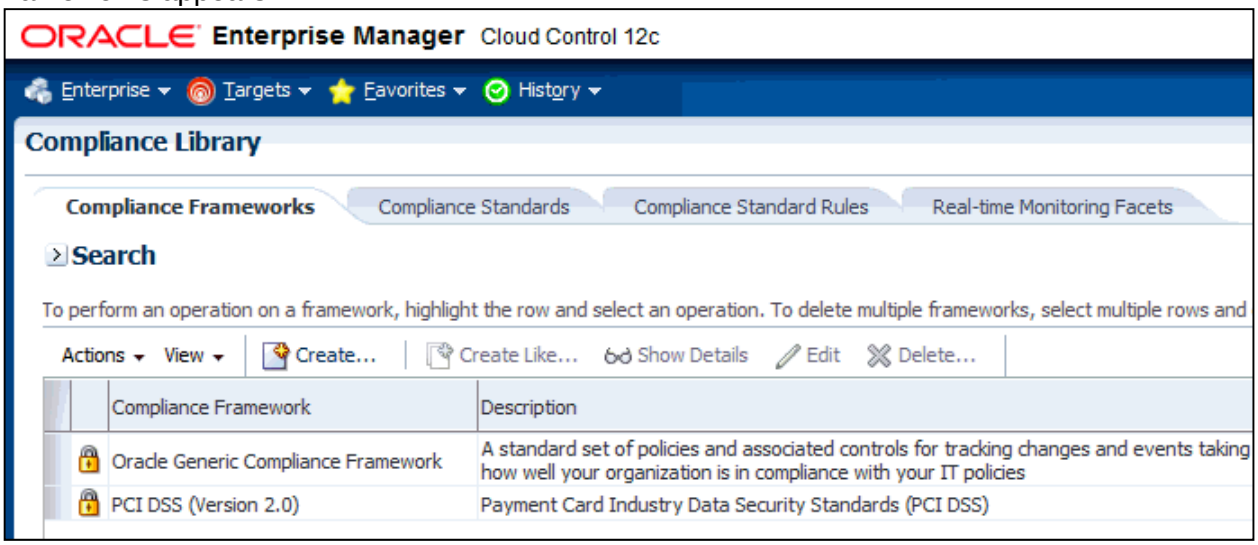
To glue all the different compliance standards for particular target types together, you use the compliance framework. The frameworks can help the administrators to create rules and standards; the compliance and security officers and auditors can take advantage of the standards and frameworks to manage compliance reports.

Tasks

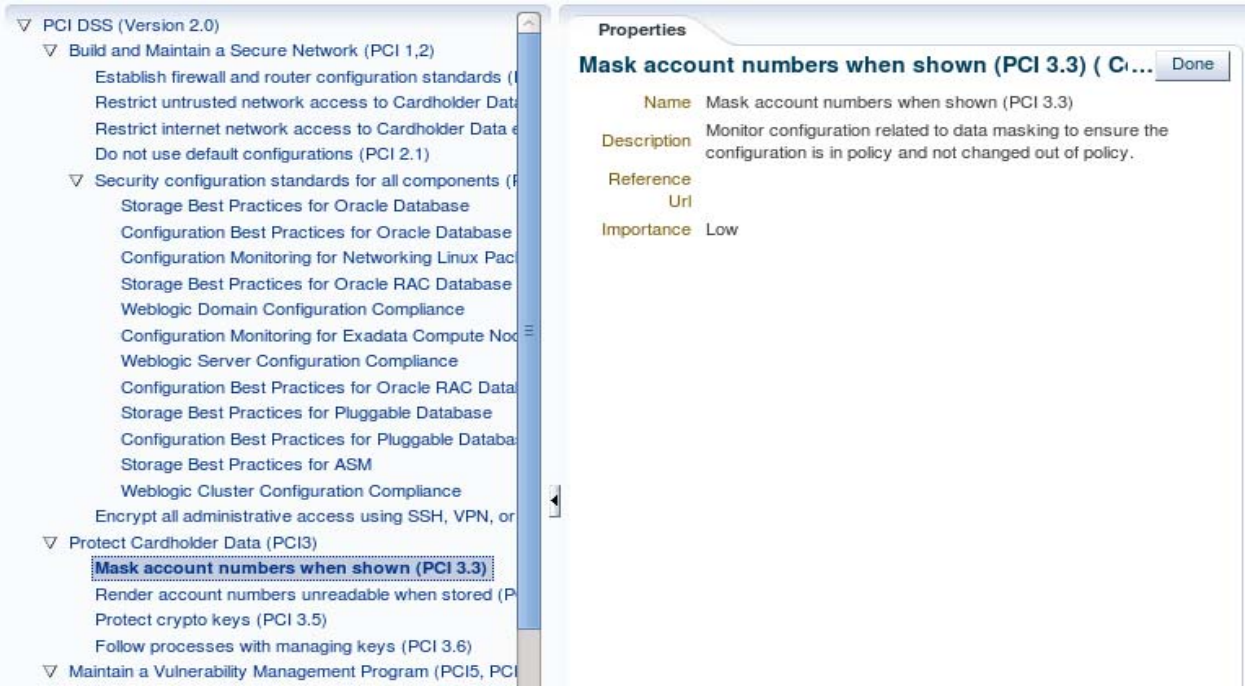
1. To review predefined compliance objects, navigate to **Enterprise > Compliance > Library**.



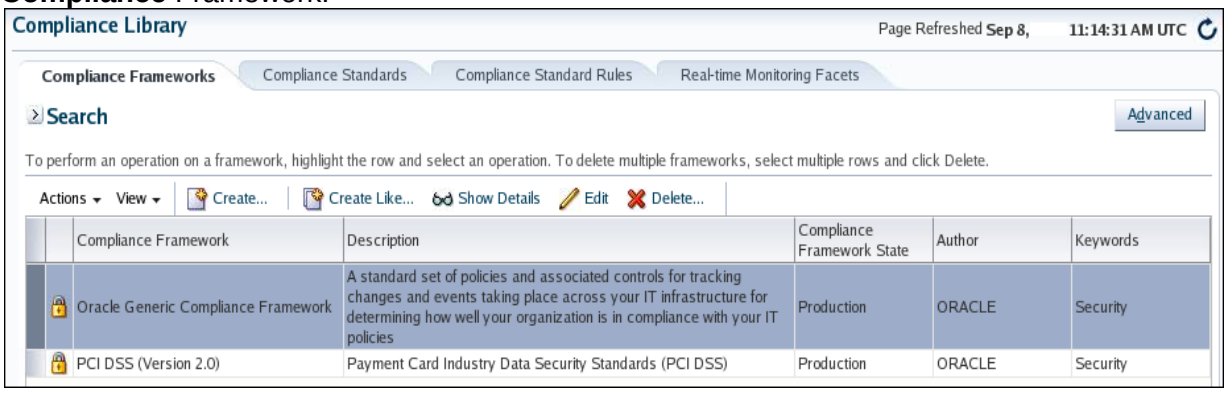
2. The Compliance Library has several tabbed pages. A list of predefined compliance frameworks appears.



3. On the Compliance Frameworks tabbed page, select a framework that interests you and click **Show Details**. Select the PCI DSS (Version 2.0) compliance framework.
4. Expand the hierarchy nodes several levels and review the descriptions; then click **Done**.

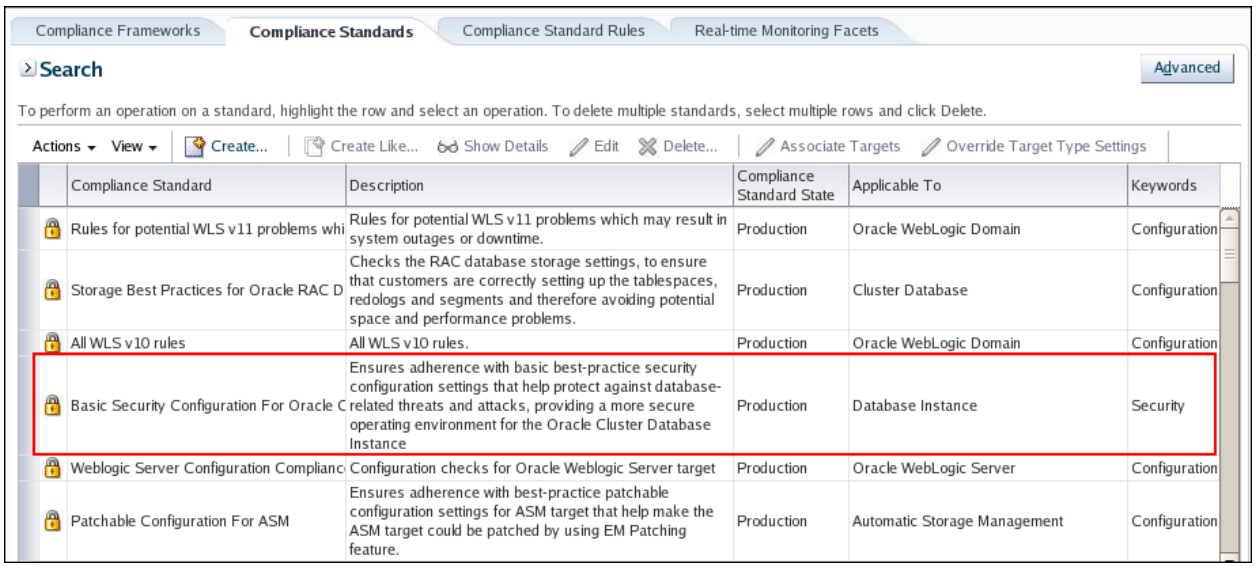


5. Back on the Compliance Frameworks tabbed page, select the **Oracle Generic Compliance** Framework.

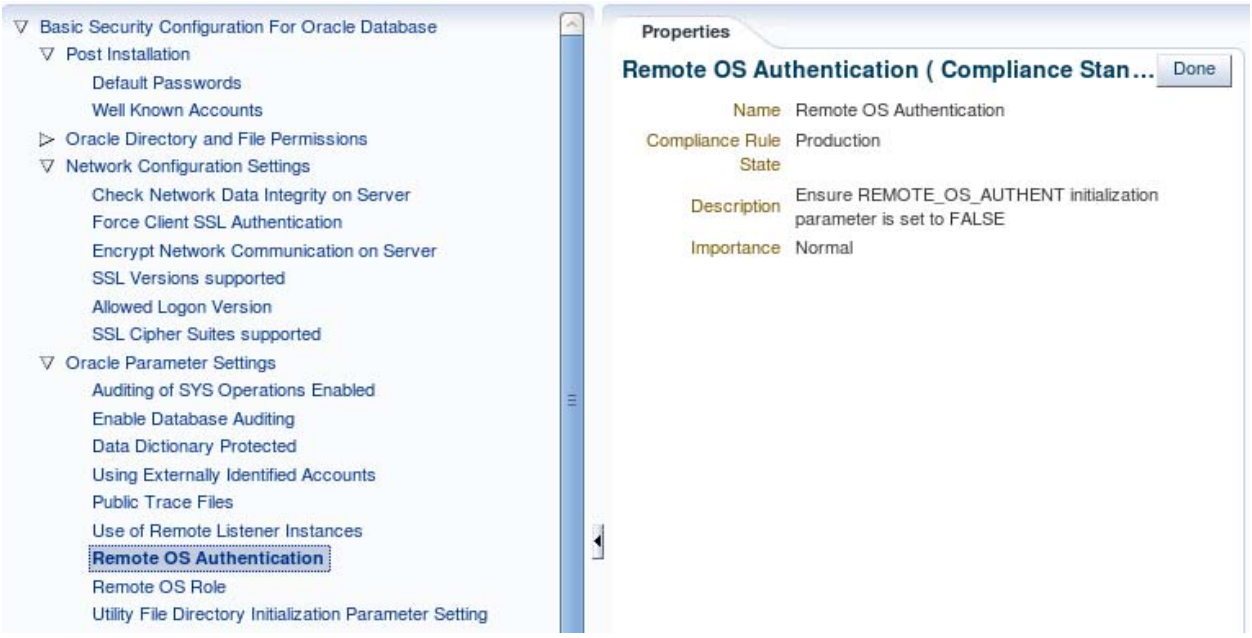


6. The hierarchy nodes displays several levels. Review the descriptions; then click D**one**.

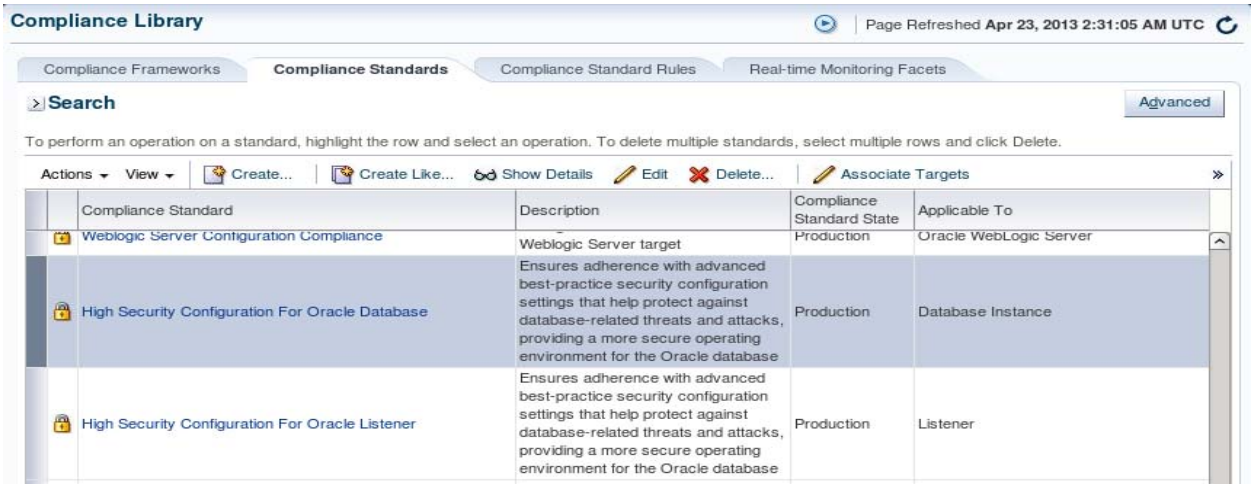
7. Click the **Compliance Standards** tab. There are quite a few standards, each for a specific target type.



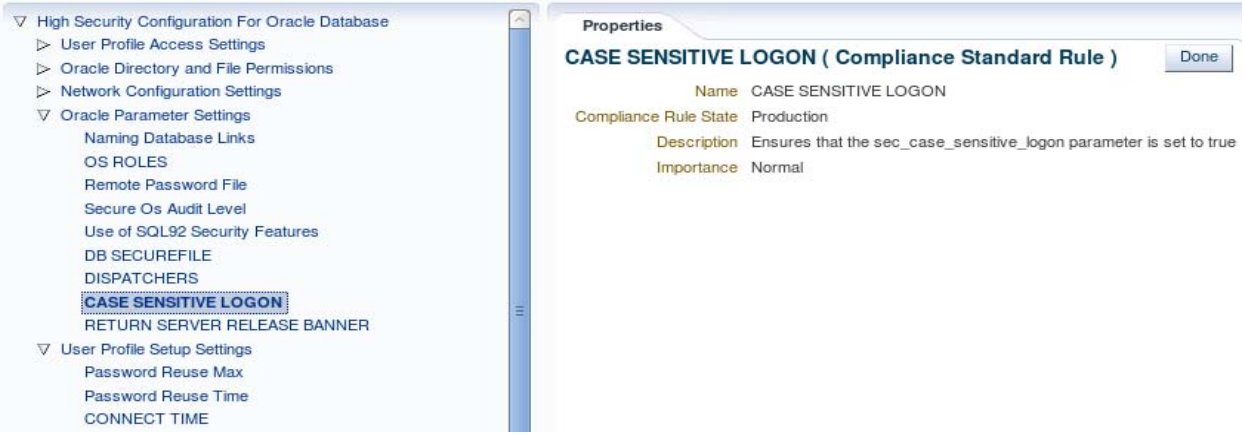
8. Review the predefined standards and then select **Basic Security Configuration For Oracle Database** (which is applicable to the Database Instance target type).
9. Expand the hierarchy node. Review the descriptions; then click D**one**.



10. Select **High Security Configuration For Oracle Database** (which is applicable to the Database Instance target type).

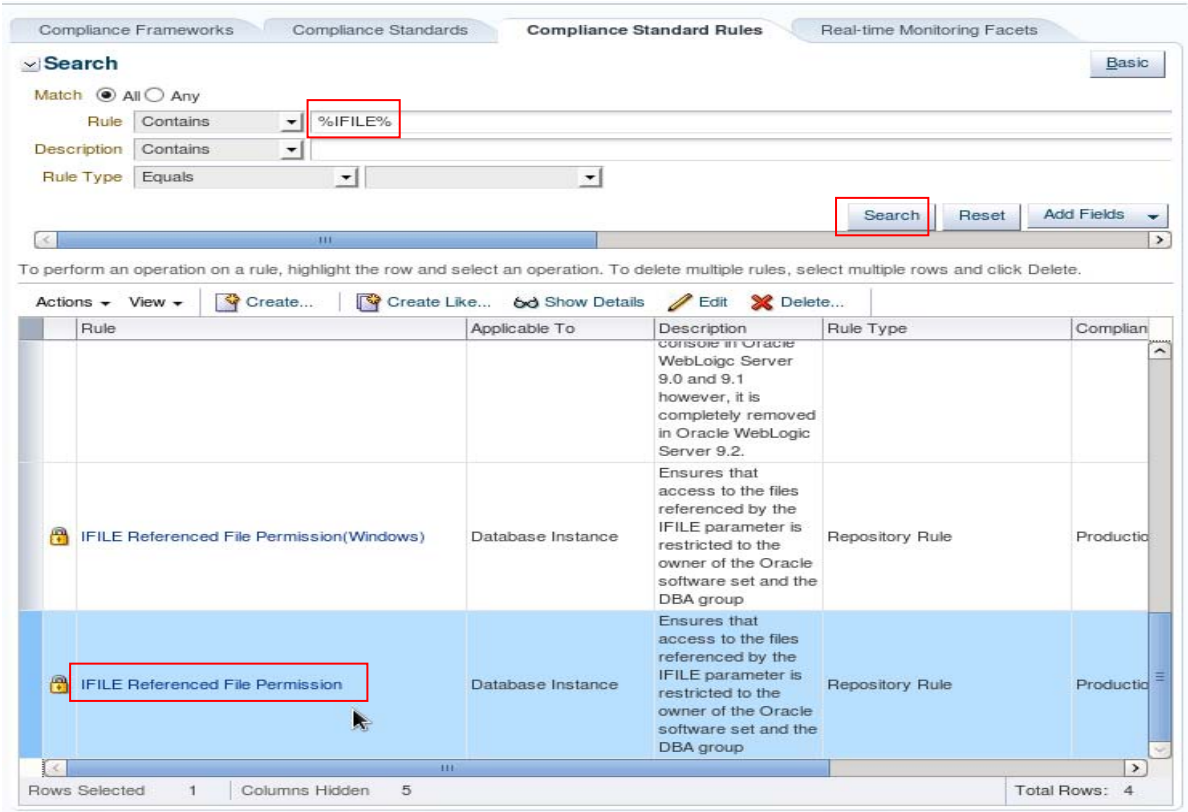


11. Review any other descriptions that may interest you, and then click **Done**.



12. Click the **Compliance Standard Rules** tab.

13. There are so many rules that you decide to use the Search functionality for finding the **IFILE Referenced File Permission**. Click the ">" icon before Search. Enter **%IFILE%** as Rule and click **Search**. Then select **IFILE Referenced File Permission**.



14. Scroll to review all details including the SQL Source of how this rule is checked in the data dictionary.

15. Click **Done** when you are finished reviewing the rule details.
16. Click **Enterprise** then **Summary** to return to the Enterprise Summary page.

Practice 3-4: Maintaining Integrity by Using Constraints

Overview

In this practice, you will use `CHECK` constraint and referential constraints to control data update and deletion.

Tasks

1. Display the existing constraints on `HR.EMPLOYEES` table in the `orcl` database.

```
$ sqlplus  hr
Enter password: ******

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
HR orcl > COL table_name         format a10
HR orcl > COL key_name           format a14
HR orcl > COL referencing_table  format a12
HR orcl > COL foreign_key_name   format a14
HR orcl > COL fk_status          format a8
HR orcl > SELECT  A.TABLE_NAME table_name,
            A.CONSTRAINT_NAME key_name,
            B.TABLE_NAME referencing_table,
            B.CONSTRAINT_NAME foreign_key_name,
            B.STATUS
     FROM  USER_CONSTRAINTS A, USER_CONSTRAINTS B
     WHERE A.CONSTRAINT_NAME = B.R_CONSTRAINT_NAME
     AND   A.TABLE_NAME = 'EMPLOYEES'
     ORDER BY 1, 2, 3, 4;
  2    3    4    5    6    7    8    9

TABLE_NAME KEY_NAME    REFERENCING_ FOREIGN_KEY_NA STATUS
---------- ------------- ------------ -------------- --------
EMPLOYEES  EMP_EMP_ID_PK  DEPARTMENTS  DEPT_MGR_FK    ENABLED
EMPLOYEES  EMP_EMP_ID_PK  EMPLOYEES    EMP_MANAGER_FK ENABLED
EMPLOYEES  EMP_EMP_ID_PK  JOB_HISTORY  JHIST_EMP_FK   ENABLED

HR orcl >
```

2. Insert a new employee in the `HR.EMPLOYEES` table as follows:

```
HR orcl > INSERT INTO hr.employees (EMPLOYEE_ID, LAST_NAME ,
          EMAIL, HIRE_DATE, JOB_ID, DEPARTMENT_ID)
```

```
          VALUES (9, 'VERRIER', 'VERRIER@test', sysdate,
                 'ST_MAN',99);
  2    3  INSERT INTO hr.employees (EMPLOYEE_ID, LAST_NAME ,
EMAIL,
*
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK) violated -
parent key not found

HR orcl >
```

The statement fails because the department does not exist. The referential constraint controls that invalid data is not inserted into the table.

3. Delete the department `30` in the `HR.DEPARTMENTS` table as follows:

```
HR orcl > DELETE FROM hr.departments WHERE department_id=30;
DELETE FROM hr.departments WHERE department_id=30
*
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK) violated -
child record found

HR orcl >
```

The statement fails because the referential constraint does not permit that the department deletion deletes all employees working in that department in cascade. The referential constraint controls that you first move the employees working in this department to another department before you can delete the department.

a. Move the employees to another department.

```
HR orcl > UPDATE hr.employees SET department_id=40
     WHERE  department_id=30;
  2
6 rows updated.

HR orcl >
```

b. Reattempt to remove the department.

```
HR orcl > DELETE FROM hr.departments WHERE department_id=30;
DELETE FROM hr.departments WHERE department_id=30
*
ERROR at line 1:
ORA-02292: integrity constraint (HR.JHIST_DEPT_FK) violated -
child record
found

HR orcl >
```

The statement fails because there is another referential constraint in another table. The `JOB_HISTORY` table contains the history of employees who had worked in that department. First remove the history records related to these employees.

```
HR orcl > DELETE FROM hr.job_history WHERE department_id=30;

6 rows deleted.

HR orcl > DELETE FROM hr.departments WHERE department_id=30;

1 row deleted.

HR orcl > ROLLBACK;

Rollback complete.

HR orcl >
```

4. Insert a new employee with a salary below the minimum legally allowed.

```
HR orcl > INSERT INTO hr.employees (EMPLOYEE_ID,
LAST_NAME,EMAIL,
                       HIRE_DATE, JOB_ID, SALARY, DEPARTMENT_ID)
          VALUES (9, 'VERRIER', 'VERRIER@test', sysdate,
                  'ST_MAN',0, 30);
  2    3    4  INSERT INTO hr.employees (EMPLOYEE_ID, LAST_NAME
     , EMAIL,
*
ERROR at line 1:
ORA-02290: check constraint (HR.EMP_SALARY_MIN) violated

HR orcl >
```

The statement fails because a `CHECK` constraint checks that the salary is higher than a minimum. Invalid value cannot be inserted into the table.

a. Examine the `HR.SALARY_MIN` constraint.

```
HR orcl > COL table_name         format a10
HR orcl > COL search_condition   format a14
HR orcl > COL constraint_name    format a18
HR orcl > SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME,
            SEARCH_CONDITION
     FROM  user_constraints
     WHERE CONSTRAINT_NAME='EMP_SALARY_MIN';
  2   3   4
CONSTRAINT_N    C TABLE_NAME SEARCH_CONDITI
--------------- - ---------- --------------
EMP_SALARY_MIN  C EMPLOYEES  salary > 0
```

```
HR orcl >
```

b. Insert a new employee with a salary above the minimum legally allowed.

```
HR orcl > INSERT INTO hr.employees (EMPLOYEE_ID, LAST_NAME ,
                EMAIL, HIRE_DATE , JOB_ID, SALARY,
                DEPARTMENT_ID)
          VALUES (9, 'VERRIER', 'VERRIER@test', sysdate,
                  'ST_MAN',500, 30);
  2    3    4
1 row created.

HR orcl > ROLLBACK;

Rollback complete.

HR orcl >
```

Practice 3-5: Maintaining Integrity by Using Triggers

Overview

In this practice, you will use triggers to maintain the stock inventory when products are sold.

Tasks

1. Find if any trigger already exist to maintain the stock inventory when products are sold.

```
HR orcl > CONNECT oe
Enter password: ******
Connected.
OE orcl > SELECT table_name, trigger_name, status, trigger_body
     FROM   user_triggers
     WHERE  trigger_name like 'STOCK%';
  2    3
no rows selected

OE orcl >
```

2. Create a simple trigger (for test purposes only) that updates the `QUANTITY_ON_HAND` in the stock when ordering product is 3515.

```
CREATE OR REPLACE TRIGGER oe.update_stock
  AFTER INSERT ON order_items
  FOR EACH ROW
  WHEN (NEW.product_id = 3515)
DECLARE
  prod_id NUMBER;
BEGIN
  prod_id := :NEW.product_id;
  UPDATE inventories
  SET    quantity_on_hand = quantity_on_hand - 100
  WHERE  product_id = prod_id;
END;
/
```

```
OE orcl > CREATE OR REPLACE TRIGGER oe.update_stock
  AFTER INSERT ON order_items
  FOR EACH ROW
  WHEN (NEW.product_id = 3515)
DECLARE
  prod_id NUMBER;
BEGIN
  prod_id := :NEW.product_id;
  UPDATE inventories
  SET    quantity_on_hand = quantity_on_hand - 100
```

```
  WHERE  product_id = prod_id;
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13
Trigger created.

OE orcl >
```

3. Display the amount of remaining items of the product ID 3515 in the stock.

```
OE orcl > SELECT QUANTITY_ON_HAND FROM OE.INVENTORIES
     WHERE PRODUCT_ID=3515;
2
QUANTITY_ON_HAND
----------------
             213

OE orcl >
```

4. Order 100 items of the product ID 3515.

```
OE orcl > INSERT INTO oe.orders (
     ORDER_ID, ORDER_DATE, CUSTOMER_ID, ORDER_TOTAL)
     VALUES (17, sysdate, 980, 100);
  2    3
1 row created.

OE orcl > INSERT INTO oe.order_items
     VALUES (17, 1, 3515, 1, 100);
  2
1 row created.

OE orcl > COMMIT;

Commit complete.

OE orcl >
```

5. Verify that the stock inventory has been updated and that the amount of remaining items of the product ID 3515 in the stock has decreased by 100.

```
OE orcl > SELECT QUANTITY_ON_HAND FROM OE.INVENTORIES
     WHERE PRODUCT_ID=3515;

QUANTITY_ON_HAND
----------------
          113
```

```
OE orcl >
```

Practice 3-6: Controlling Data Access by Using Views

Overview

In this practice, you create different views based on the `HR.EMPLOYEES` and `HR.DEPARTMENTS` tables displaying selected rows and columns according to the user's role in the company. `JIM`, the `HR` assistant, should be able to view all information of any employee except those of the managers. TOM should only be allowed to view the first and last names, and the department ID and name where any employee works.

Tasks

1. Create the `HR_ASSISTANT` view.

```
OE orcl > CONNECT HR
Enter password: ******
Connected.
HR orcl > CREATE VIEW hr_assistant
     AS SELECT * FROM HR.EMPLOYEES
        WHERE job_id NOT IN
        (SELECT job_id FROM HR.JOBS WHERE
job_title='President');
2    3    4
View created.

HR orcl > GRANT SELECT ON hr.hr_assistant TO jim;

Grant succeeded.

HR orcl >
```

2. Create the `HR_CLERK` view.

```
HR orcl > CREATE VIEW hr_clerk
     AS SELECT first_name, last_name, department_name
        FROM   hr.employees e, hr.departments d
        WHERE  e.DEPARTMENT_ID = d.DEPARTMENT_ID;
2    3    4
View created.

HR orcl > GRANT SELECT ON hr.hr_clerk TO tom, jim;

Grant succeeded.

HR orcl >
```

3. Verify that only `JIM` can view all information of any employees except the president, and that `TOM` can only view some information of the employees.

```
HR orcl > CONNECT jim
Enter password: ******
```

```
Connected.
JIM orcl > SELECT * FROM hr.employees;
SELECT * FROM hr.employees
                 *
ERROR at line 1:
ORA-00942: table or view does not exist

JIM orcl > SELECT count(*) FROM hr.hr_assistant;

  COUNT(*)
----------
       106

JIM orcl >
```

Notice that the view returns 106 rows and not 107 rows.

```
JIM orcl > CONNECT tom
Enter password: ******
Connected.
TOM orcl > SELECT * FROM hr.employees;
SELECT * FROM hr.employees
                 *
ERROR at line 1:
ORA-00942: table or view does not exist

TOM orcl > SELECT * FROM hr.hr_assistant;
SELECT * FROM hr.employees
                 *
ERROR at line 1:
ORA-00942: table or view does not exist

TOM orcl > SELECT * FROM hr.hr_clerk WHERE
last_name='Greenberg';

FIRST_NAME           LAST_NAME                DEPARTMENT_NAME
-------------------- ------------------------ ------------------
Nancy                Greenberg                Finance

TOM orcl > SELECT salary FROM hr.hr_clerk
           WHERE last_name='Greenberg';
SELECT salary FROM hr.hr_clerk WHERE last_name='Greenberg'
       *
ERROR at line 1:
```

```
ORA-00904: "SALARY": invalid identifier

TOM orcl >
```

4. Drop the views.

```
TOM orcl > CONNECT hr
Enter password: ******
Connected.
HR orcl > DROP VIEW hr.hr_assistant;

View dropped.

HR orcl > DROP VIEW hr.hr_clerk;

View dropped.

HR orcl > EXIT
$
```

.

Practice 3-7: Using Database Vault Realms to Disallow Access to Objects

Overview

In this practice, you will verify that Database Vault realms configuration can disallow `HR` from viewing any data in his own schema objects, protecting objects from any user being granted system and or object privileges.

Tasks

1. Make sure you are in the `~/labs/DV` directory and your environment points to the `orcl` instance.

```
$ cd ~/labs/DV
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

2. Run the `DV_setup.sh` script to configure Database Vault in the database. This may take several minutes to complete.

```
$ ./DV_setup.sh

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

drop user sec_admin cascade
          *
ERROR at line 1:
ORA-01918: user 'SEC_ADMIN' does not exist



drop user accts_admin cascade
          *
ERROR at line 1:
ORA-01918: user 'ACCTS_ADMIN' does not exist



User created.

User created.
```

```
Grant succeeded.

PL/SQL procedure successfully completed.

Connected.

PL/SQL procedure successfully completed.
$
```

3. Restart the instance.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SYS orcl > shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS orcl > startup
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289400 bytes
Variable Size             264241416 bytes
Database Buffers          226492416 bytes
Redo Buffers                8036352 bytes
Database mounted.
Database opened.
SYS orcl > exit
$
```

4. Run the `DV_create_realm.sh` script to create a Database Vault realm protecting the `HR.EMPLOYEES` and `HR.DEPARTMENTS` tables from any access, even from `HR` access.

```
$ ./DV_create_realm.sh

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Oracle Database Vault and Real Application Testing
options
```

```
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
$
```

Notice that the banner shows the **Oracle Database Vault** option enabled.

5. Connect as HR to verify that HR does not have any access to the HR.EMPLOYEES and HR.DEPARTMENTS tables.

```
$ sqlplus hr
Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Oracle Database Vault and Real Application Testing
options

HR orcl >
HR orcl > select * from hr.employees;
select * from hr.employees
                 *
ERROR at line 1:
ORA-01031: insufficient privileges

HR orcl > select * from hr.departments;
select * from hr.departments
                 *
ERROR at line 1:
ORA-01031: insufficient privileges

HR orcl >
```

6. Verify that HR can access to other tables owned in his schema.

```
HR orcl > select * from hr.jobs;

JOB_ID     JOB_TITLE                          MIN_SALARY MAX_SALARY
---------- ------------------------------- ---------- ---------
AD_PRES    President                              20080      40000
AD_VP      Administration Vice President           15000      30000
AD_ASST    Administration Assistant                 3000      6000
FI_MGR     Finance Manager                          8200      16000
FI_ACCOUNT Accountant                                    4200
9000
```

```
AC_MGR     Accounting Manager                    8200      16000
AC_ACCOUNT Public Accountant                     4200      9000
SA_MAN     Sales Manager                        10000      20080
SA_REP     Sales Representative                       6000
12008
PU_MAN     Purchasing Manager                   8000      15000
PU_CLERK   Purchasing Clerk                     2500       5500
ST_MAN     Stock Manager                         5500      8500
ST_CLERK   Stock Clerk                          2008       5000
SH_CLERK   Shipping Clerk                        2500      5500
IT_PROG    Programmer                                4000
10000
MK_MAN     Marketing Manager                    9000      15000
MK_REP     Marketing Representative             4000       9000
HR_REP     Human Resources Representative       4000       9000
PR_REP     Public Relations Representative      4500      10500

19 rows selected.

HR orcl >
```

7. Select from a non-existing table.

```
HR orcl > select * from hr.test_tab;
select * from hr.test_tab
                 *
ERROR at line 1:
ORA-00942: table or view does not exist

HR orcl > EXIT
$
```

The error message is not the same as in task 5 or task 6.

8. Run the `DV_drop_realm.sh` script to remove the Database Vault protection on the `HR.EMPLOYEES` and `HR.DEPARTMENTS` tables.

```
$ ./DV_drop_realm.sh

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Oracle Database Vault and Real Application Testing
options

PL/SQL procedure successfully completed.
$
```

9. Run the `DV_disable.sh` script to disable Database Vault in the database.

```
$ ./DV_disable.sh

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Oracle Database Vault and Real Application Testing
options

Connected.

PL/SQL procedure successfully completed.
$
```

10. Restart the instance.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Oracle Database Vault and Real Application Testing
options

SYS orcl > shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS orcl > startup
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289400 bytes
Variable Size             264241416 bytes
Database Buffers          226492416 bytes
Redo Buffers                8036352 bytes
Database mounted.
Database opened.
SYS orcl > exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
```

```
$
```

Notice that the banner does not show the **Oracle Database Vault** option anymore. It is disabled.

11. Verify the HR can view the tables he is the owner of.

```
$ sqlplus hr

Enter password: ******

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

HR orcl > SELECT count(*) FROM hr.employees;

  COUNT(*)
----------
       107

HR orcl > exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
$
```

Practices for Lesson 4: Implementing Basic Database Security

Chapter 4

Practices for Lesson 4: Overview

Practices Overview

In these practices, you will implement the basic database security features and investigate if your databases are compliant with the Basic Security Configuration For Oracle Database compliance standards.

Note: From now on, in the following practices, the SQL prompt will be displayed with the default value “SQL>” to make the practice documents reading easier.

Practice 4-1: Creating the Security Officer Account

Overview

In this practice, you will create the security officer account that has privileges to create user accounts, grant privileges, and administer fine-grained auditing and fine-grained access control in the `orcl` database.

In this and subsequent practices, security is administered by a single user. Be sure to use this account whenever possible.

Tasks

1. Connect as `SYSTEM` in `orcl` instance to create the `SEC` user, giving it the following properties:
   - Name is `SEC`
   - Password is `oracle_4sec`
   - This user must be able to allocate space in the `USERS` tablespace for security related tables, and objects
   - Can create a session and grant the privilege to other users to create a session
   - Can select from any table in the database, including the SYS schema
   - Can create or drop any context in the database
   - Can create, alter, and drop users
   - Can create roles and can alter and drop any roles
   - Can create tables, procedures, and triggers (including the `ADMINISTER DATABASE TRIGGER` privilege, which allows the user to create database triggers)
   - Can administer OS file access through DIRECTORY objects
   - Can administer profiles
   - Can execute audit commands
   - Can execute `ALTER SYSTEM` commands (allows the user to change initialization parameters)
   - Can grant and revoke any object privilege
   - Can execute `DBMS_SESSION`. This privilege is granted from the `SYS` user to `PUBLIC` by default

   a. Use the `oraenv` utility to set the `ORACLE_SID` environment variable to the `orcl` value.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

   b. Execute the `create_sec.sh` script. Make sure you are in the `~/labs/USERS` directory.

```
$ cd ~/labs/USERS
$ ./create_sec.sh
```

```
SQL*Plus: Release 12.1.0.1.0 Production on Thu Jun 13 23:07:05
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> DROP USER sec CASCADE;
DROP USER sec CASCADE
          *
ERROR at line 1:
ORA-01918: user 'SEC' does not exist


SQL> CREATE USER sec IDENTIFIED BY oracle_4sec
  2  DEFAULT TABLESPACE USERS
  3  QUOTA UNLIMITED ON  USERS;

User created.

SQL>
SQL> GRANT create session
  2    TO sec
  3    WITH ADMIN OPTION;

Grant succeeded.

SQL>
SQL> GRANT select_catalog_role, select any table,
  2        create any context, drop any context,
  3        create user, alter user, drop user,
  4        create role, alter any role, drop any role,
  5        create table, create procedure,
  6        create any trigger, administer database trigger,
  7        create any directory, alter profile, create profile,
  8        drop profile, audit system, alter system,
  9        grant any object privilege, grant any privilege,
grant any role
```

```
 10    TO sec;

Grant succeeded.

SQL>
SQL> GRANT execute on DBMS_SESSION to sec;

Grant succeeded.

SQL> GRANT execute on UTL_FILE to sec;

Grant succeeded.

SQL>
SQL> EXIT
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
$
```

2. The security officer immediately takes some actions due to basic security issues.
   a. Sample schema accounts `HR`, `OE`, `SH`, `PM`, `BI`, and `IX` are well known; they should not be installed unless needed. If they are not needed, the passwords should be expired and the accounts locked when not being used. After a password is marked as expired, the password must be changed before the account can be used again.

```
$ sqlplus sec
Enter password: *******

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
SQL> ALTER USER PM PASSWORD EXPIRE ACCOUNT LOCK;

User altered.

SQL> ALTER USER BI PASSWORD EXPIRE ACCOUNT LOCK;

User altered.

SQL> ALTER USER IX PASSWORD EXPIRE ACCOUNT LOCK;
```

```
User altered.

SQL>
```

b. Because it is dangerous to work with `UTL_FILE_DIR` parameter set to *, you reset the `UTL_FILE_DIR` parameter to `NULL`, so that no one can read from or write to any directory using the `UTL_FILE` package. Then you configure the database so that users can write to the `/home/oracle/student` directory:

1) Reset the `UTL_FILE_DIR` parameter to `NULL`.

```
SQL> ALTER SYSTEM SET utl_file_dir='' SCOPE=spfile;

System altered.

SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2290024 bytes
Variable Size             264244888 bytes
Database Buffers          226492416 bytes
Redo Buffers                8032256 bytes
Database mounted.
Database opened.
SQL>
```

2) Configure the database to allow writes using the `DIRECTORY` objects. Create the `/home/oracle/student` directory on the OS. Create a directory object for the `/home/oracle/student` directory. You can later grant `READ` or `WRITE` privileges to the directory to certain users.

```
SQL> !mkdir /home/oracle/student

SQL> CONNECT sec
Enter password: *******
Connected.
SQL> CREATE DIRECTORY student AS '/home/oracle/student';

Directory created.
```

```
SQL>
```

3) Test the configuration. The following PL/SQL block writes the current database time to the `db_time.lst` file. The PL/SQL block accepts a single parameter: the uppercase name of the directory object that you want to write to (`STUDENT`).

```
SQL> DECLARE
  file_handle          UTL_FILE.FILE_TYPE;
  file_mode            VARCHAR2(1) := 'w';
  file_name            VARCHAR2(15) := 'db_time.lst';
  file_location        VARCHAR2(80) := '&1';
  file_data            VARCHAR2(100);
  BEGIN
  file_handle := utl_file.fopen(file_location, file_name,
file_mode);
  IF utl_file.is_open(file_handle) THEN
     file_data := current_timestamp;
     utl_file.put(file_handle, file_data);
     utl_file.fclose(file_handle);
  ELSE
     dbms_output.put_line('The file was not opened.');
  END IF;
 END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16   17  Enter value for 1: /home/oracle
old   5:   file_location  VARCHAR2(80) := '&1';
new   5:   file_location  VARCHAR2(80) := '/home/oracle';
DECLARE
*
ERROR at line 1:
ORA-29280: invalid directory path
ORA-06512: at "SYS.UTL_FILE", line 41
ORA-06512: at "SYS.UTL_FILE", line 478
ORA-06512: at line 8

SQL>
```

Notice the error. The `/home/oracle` OS directory is not a directory object defined in the database. Use a directory defined in the database.

```
SQL> DECLARE
  file_handle          UTL_FILE.FILE_TYPE;
  file_mode            VARCHAR2(1) := 'w';
  file_name            VARCHAR2(15) := 'db_time.lst';
  file_location        VARCHAR2(80) := '&1';
  file_data            VARCHAR2(100);
```

```
  BEGIN
  file_handle := utl_file.fopen(file_location, file_name,
file_mode);
  IF utl_file.is_open(file_handle) THEN
     file_data := current_timestamp;
     utl_file.put(file_handle, file_data);
     utl_file.fclose(file_handle);
  ELSE
     dbms_output.put_line('The file was not opened.');
  END IF;
 END;
/
Enter value for 1: STUDENT
old   5:    file_location           VARCHAR2(80) := '&1';
new   5:    file_location           VARCHAR2(80) := 'STUDENT';

PL/SQL procedure successfully completed.

SQL>
```

4) Verify that the `db_time.lst` file is written to the directory after executing the PL/SQL block.

```
SQL> HOST cat /home/oracle/student/db_time.lst
05-JUL-13 10.01.49.700632000 AM +00:00
SQL>
```

c. Do any users in your database have the `DBA` role, `SYSOPER`, `SYSDBA`, `SYSKM`, `SYSDG`, or `SYSBACKUP` privilege that they do not need? Fix this problem.

1) Find users who are granted the `DBA` role by querying the `DBA_ROLE_PRIVS` view.

```
SQL> COL grantee  FORMAT a12
SQL> COL granted_role FORMAT a12
SQL> SELECT * FROM dba_role_privs  WHERE granted_role='DBA';

GRANTEE              GRANTED_ROLE           ADM DEF COM
-------------------- ---------------------- --- --- ---
SYS                  DBA                    YES YES  NO
SCOTT                DBA                    NO  YES  NO
SYSTEM               DBA                    YES YES YES

SQL>
```

2) `SCOTT` has no need for the `DBA` role because this is a demo account that has been locked and the password expired. Revoke the `DBA` role from `SCOTT`. To revoke a role, you must have been granted the role with `ADMIN OPTION`. You can revoke any role if you have the `GRANT ANY ROLE` system privilege.

```
SQL> REVOKE DBA FROM scott;

Revoke succeeded.

SQL> SELECT * FROM dba_role_privs  WHERE granted_role='DBA';

GRANTEE              GRANTED_ROLE           ADM DEF COM
-------------------- ---------------------- --- --- ---
SYS                  DBA                    YES YES  NO
SYSTEM               DBA                    YES YES YES

SQL>
```

d. The users with the `SYSDBA` or `SYSOPER` privilege are listed in the oracle password file. `SCOTT` and `HR` have no need for these privileges. Only `SYSDBA` can `GRANT` or `REVOKE` these privileges.

```
SQL> COL username  FORMAT a12
SQL> SELECT * FROM v$pwfile_users;

USERNAME     SYSDB SYSOP SYSAS SYSBA SYSDG SYSKM     CON_ID
------------ ----- ----- ----- ----- ----- ----- ----------
SYS          TRUE  TRUE  FALSE FALSE FALSE FALSE         0
SYSDG        FALSE FALSE FALSE FALSE TRUE  FALSE         0
SYSBACKUP    FALSE FALSE FALSE TRUE  FALSE FALSE               0
SYSKM        FALSE FALSE FALSE FALSE FALSE TRUE          0
SCOTT        TRUE  FALSE FALSE FALSE FALSE FALSE         0
HR           FALSE TRUE  FALSE FALSE FALSE FALSE         0

6 rows selected.

SQL> REVOKE SYSOPER FROM hr;
REVOKE SYSOPER FROM hr
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> CONNECT / AS SYSDBA
Connected.
SQL> REVOKE SYSOPER FROM hr;

Revoke succeeded.

SQL> REVOKE SYSDBA FROM scott;
```

```
Revoke succeeded.

SQL>
SQL> SELECT * FROM v$pwfile_users;

USERNAME     SYSDB SYSOP SYSAS SYSBA SYSDG SYSKM    CON_ID
------------ ----- ----- ----- ----- ----- ----- ---------
SYS          TRUE  TRUE  FALSE FALSE FALSE FALSE      0
SYSDG        FALSE FALSE FALSE FALSE TRUE  FALSE      0
SYSBACKUP    FALSE FALSE FALSE TRUE  FALSE FALSE           0
SYSKM        FALSE FALSE FALSE FALSE FALSE TRUE            0

SQL>
```

1) Do any users in your database have the RESOURCE role? If there are some users being granted the RESOURCE role, check that the UNLIMITED TABLESPACE system privilege is not granted. In Oracle Database 12c, the RESOURCE role is not granted the UNLIMITED TABLESPACE system privilege anymore.

```
SQL> CONNECT sec
Enter password: *******
Connected.
SQL> SELECT grantee, privilege, granted_role
     FROM   dba_sys_privs JOIN dba_role_privs USING (grantee)
     WHERE  granted_role='RESOURCE'
     AND    privilege = 'UNLIMITED TABLESPACE';

GRANTEE        PRIVILEGE              GRANTED_ROLE
-------------- --------------------   ---------------------
HR             UNLIMITED TABLESPACE   RESOURCE
OE             UNLIMITED TABLESPACE   RESOURCE
BI             UNLIMITED TABLESPACE   RESOURCE
IX             UNLIMITED TABLESPACE   RESOURCE
SH             UNLIMITED TABLESPACE   RESOURCE
PM             UNLIMITED TABLESPACE   RESOURCE
XDB            UNLIMITED TABLESPACE   RESOURCE
OJVMSYS        UNLIMITED TABLESPACE   RESOURCE
MDSYS          UNLIMITED TABLESPACE   RESOURCE
APEX_040200    UNLIMITED TABLESPACE   RESOURCE
SYS            UNLIMITED TABLESPACE   RESOURCE
OUTLN          UNLIMITED TABLESPACE   RESOURCE
CTXSYS         UNLIMITED TABLESPACE   RESOURCE
DVSYS          UNLIMITED TABLESPACE   RESOURCE
LBACSYS        UNLIMITED TABLESPACE   RESOURCE
```

```
15 rows selected.

SQL>
```

2) Find other users who may be granted the `UNLIMITED TABLESPACE` privilege by querying the `DBA_SYS_PRIVS` view.

```
SQL> SELECT grantee FROM dba_sys_privs
     WHERE  privilege = 'UNLIMITED TABLESPACE'
```

```
     AND    grantee NOT IN (SELECT grantee
     FROM   dba_sys_privs JOIN dba_role_privs USING (grantee)
     WHERE  granted_role='RESOURCE'
```

```
     AND    privilege = 'UNLIMITED TABLESPACE');
```

```
  2    3    4    5    6
GRANTEE
--------------------
TOM
SI_INFORMTN_SCHEMA
WMSYS
DBSNMP
ORDSYS
ORDDATA
SYSTEM
SYSBACKUP

8 rows selected.

SQL>
```

3) If necessary, revoke the `UNLIMITED TABLESPACE` privilege from `TOM` user.

```
SQL> REVOKE unlimited tablespace FROM tom;

Revoke succeeded.

SQL> EXIT
$
```

Practice 4-2: Managing Secure Passwords

Overview

In this practice, the security officer will ensure that the use of simple passwords is not possible and that all users will follow strong password management rules. Oracle Database 12*c* provides password management by default with one of the three password verification function effective by default.

Tasks

1. Determine what limits are applied with the `DEFAULT` profile. Then, set up password management by performing the following steps:
   a. List the rows related to password management from the current profiles in the system. Use the `SEC` account. Save the command that you use.

```
$ sqlplus sec
Enter password: ******

Last Successful login time: Tue May 21 2013 03:58:51 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> set pagesize 40
```

```
SQL> col profile format A10
SQL> col limit format A22
SQL> col resource_name format A25
SQL> SELECT profile, resource_name, limit
     FROM dba_profiles
     WHERE PROFILE = 'DEFAULT'
     AND resource_type = 'PASSWORD';
  2    3    4
PROFILE    RESOURCE_NAME             LIMIT
---------- ------------------------- ----------------------
DEFAULT    FAILED_LOGIN_ATTEMPTS      10
DEFAULT    PASSWORD_LIFE_TIME         180
DEFAULT    PASSWORD_REUSE_TIME        UNLIMITED
DEFAULT    PASSWORD_REUSE_MAX         UNLIMITED
DEFAULT    PASSWORD_VERIFY_FUNCTION   NULL
DEFAULT    PASSWORD_LOCK_TIME         1
DEFAULT    PASSWORD_GRACE_TIME        7

7 rows selected.
```

```
SQL> SAVE $HOME/labs/default_profile.sql REPLACE
Wrote file /home/oracle/labs/default_profile.sql
SQL> EXIT
$
```

b. Because the password verification function must be owned by `SYS`, connect as the `SYS` user and verify that the default profile is assigned to all users to apply one of the three available password verification functions. Read each of them and choose the strongest one. The script explains in the last part how to apply one of the three verify functions to the `DEFAULT` profile.

```
$ cd $ORACLE_HOME/rdbms/admin
$ cat utlpwdmg.sql
…
Rem Function: "ora12c_verify_function" - provided from 12c
onwards
Rem
Rem This function makes the minimum complexity checks like
Rem the minimum length of the password, password not same as the
Rem username, etc. The user may enhance this function according
to
Rem the need.
Rem This function must be created in SYS schema.
Rem connect sys/<password> as sysdba before running the script

CREATE OR REPLACE FUNCTION ora12c_verify_function
(username varchar2,
 password varchar2,
 old_password varchar2)
…
Rem Function: "ora12c_strong_verify_function" - provided from12c
onwards for
Rem           stringent password check requirements.
Rem
Rem This function is provided to give stronger password
complexity function
Rem that would take into consideration recommendations from
Department of
Rem Defense Database Security Technical Implementation Guide.

CREATE OR REPLACE FUNCTION ora12c_strong_verify_function
(username varchar2,
 password varchar2,
 old_password varchar2)
RETURN boolean IS
   differ integer;
```

```
…
Rem Function: "verify_function_11G" - provided from 11G onwards.
Rem
Rem This function makes the minimum complexity checks like
Rem the minimum length of the password, password not same as the
Rem username, etc. The user may enhance this function according
to
Rem the need.

CREATE OR REPLACE FUNCTION verify_function_11G
(username varchar2,
 password varchar2,
 old_password varchar2)
…
-- This script alters the default parameters for Password
Management
-- This means that all the users on the system have Password
Management
-- enabled and set to the following values unless another
profile is
-- created with parameter values set to different value or
UNLIMITED
-- is created and assigned to the user.

ALTER PROFILE DEFAULT LIMIT
PASSWORD_LIFE_TIME 180
PASSWORD_GRACE_TIME 7
PASSWORD_REUSE_TIME UNLIMITED
PASSWORD_REUSE_MAX  UNLIMITED
FAILED_LOGIN_ATTEMPTS 10
PASSWORD_LOCK_TIME 1
PASSWORD_VERIFY_FUNCTION ora12c_verify_function;

/**
The below set of password profile parameters would take into
consideration
recommendations from Center for Internet Security[CIS Oracle
11g].

ALTER PROFILE DEFAULT LIMIT
PASSWORD_LIFE_TIME 90
PASSWORD_GRACE_TIME 3
PASSWORD_REUSE_TIME 365
PASSWORD_REUSE_MAX  20
```

```
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1
PASSWORD_VERIFY_FUNCTION ora12c_verify_function;
*/

/**
The below set of password profile parameters would take into
consideration recommendations from Department of Defense
Database
Security Technical Implementation Guide[STIG v8R1].

ALTER PROFILE DEFAULT LIMIT
PASSWORD_LIFE_TIME 60
PASSWORD_REUSE_TIME 365
PASSWORD_REUSE_MAX  5
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_VERIFY_FUNCTION ora12c_strong_verify_function;
$
```

c. Using SQL*Plus, connect to the database `AS SYSDBA` and verify that the three password verification functions are not created yet.

```
$ sqlplus / AS SYSDBA

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> SET ECHO ON
SQL> SELECT object_name, object_type
     FROM dba_objects
     WHERE object_name LIKE '%VERIFY_FUNCTION%';
  2   3
no rows selected.

SQL> SELECT LIMIT from dba_profiles
     where profile = 'DEFAULT'
     and   resource_name = 'PASSWORD_VERIFY_FUNCTION';
2
LIMIT
----------------------------------------------------------------
NULL

SQL>
```

Note: If the database had been created with DBCA, the DEFAULT profile would have the `PASSWORD_VERIFY_FUNCTION` limit set to `ora12c_verify_function` function.

d. Alter the `DEFAULT` profile to apply the strong password verification function chosen in task b. Be aware that all new accounts will be under the rules of the new password verify function. If you do not want this situation, create a profile and assign another password verify function to the new profile. This allows you to keep the `DEFAULT` profile with the basic password verify function.

1) Create the functions.

```
SQL> @$ORACLE_HOME/rdbms/admin/utlpwdmg.sql

Function created.


Function created.


Function created.


Grant succeeded.


Function created.


Grant succeeded.


Function created.


Grant succeeded.


Function created.


Grant succeeded.


Profile altered.
```

The output has been modified to show only the results.

2) Verify that the password verify functions are created.

```
SQL> col OBJECT_NAME format A38
SQL> col OBJECT_TYPE  format A20
SQL> SELECT object_name, object_type
     FROM dba_objects
     WHERE object_name LIKE '%VERIFY_FUNCTION%';
  2    3

OBJECT_NAME                                OBJECT_TYPE
---------------------------------------- --------------------
ORA12C_VERIFY_FUNCTION                     FUNCTION
ORA12C_STRONG_VERIFY_FUNCTION              FUNCTION
VERIFY_FUNCTION_11G                        FUNCTION
VERIFY_FUNCTION                        FUNCTION

SQL>
```

3) Update the DEFAULT profile with the password verify function.

```
SQL> ALTER PROFILE default LIMIT
      PASSWORD_VERIFY_FUNCTION ora12c_strong_verify_function;
2
Profile altered.

SQL>
```

e. View the changes applied. Repeat the command from step 2a as the SEC user and note the differences.

```
SQL> CONNECT SEC
Enter password: ******
Connected.
SQL> COL profile format A7
SQL> COL resource_name format A32
SQL> COL limit format A30
SQL> @$HOME/labs/default_profile.sql
SQL> SELECT profile, resource_name, limit
     FROM dba_profiles
     WHERE PROFILE = 'DEFAULT'
     AND resource_type = 'PASSWORD';

PROFILE  RESOURCE_NAME            LIMIT
------- ------------------------ -----------------------
DEFAULT FAILED_LOGIN_ATTEMPTS     10
DEFAULT PASSWORD_LIFE_TIME  180
DEFAULT PASSWORD_REUSE_TIME       UNLIMITED
DEFAULT PASSWORD_REUSE_MAX        UNLIMITED
```

```
DEFAULT PASSWORD_VERIFY_FUNCTION ORA12C_STRONG_VERIFY_FUNCTION
DEFAULT PASSWORD_LOCK_TIME        1
DEFAULT PASSWORD_GRACE_TIME       7

7 rows selected.

SQL>
```

2. Create a user and verify that the password is secure with the verify function applied in the profile.

```
SQL> CREATE USER ann IDENTIFIED BY xxx12345;
CREATE USER ann IDENTIFIED BY xxx12345
*
ERROR at line 1:
ORA-28003: password verification for the specified password
failed
ORA-20001: Password length less than 9

SQL> CREATE USER ann IDENTIFIED BY A_xxx12345667890???!!!_yyy;
CREATE USER ann IDENTIFIED BY A_xxx12345667890???!!!_yyy
                                                *
ERROR at line 1:
ORA-00911: invalid character

SQL> CREATE USER ann IDENTIFIED BY A_xxx12345667890_yyy;
CREATE USER ann IDENTIFIED BY A_xxx12345667890_yyy
*
ERROR at line 1:
ORA-28003: password verification for the specified password
failed
ORA-20023: Password must contain at least 2 uppercase
character(s)

SQL> CREATE USER ann IDENTIFIED BY A_xxx12345667890_Yyy;

User created.

SQL>
```

3. What happens to the `SYS` user when he alters his own password?

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER USER sys IDENTIFIED BY oracle_4U;
```

```
User altered.

SQL>
```

Notice that SYS is not under the rules of any password checking function even if defined in the DEFAULT profile.

4. What happens to a user being granted the SYSDBA privilege when he alters his own password?

```
SQL> GRANT sysdba TO tom;

Grant succeeded.

SQL> CONNECT tom AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER USER tom IDENTIFIED BY oracle_4U;
ALTER USER tom IDENTIFIED BY oracle_4U
*
ERROR at line 1:
ORA-28003: password verification for the specified password
failed
ORA-20023: Password must contain at least 2 uppercase
character(s)

SQL> ALTER USER tom IDENTIFIED BY Strong_pass_6W;
ALTER USER tom IDENTIFIED BY Strong_pass_6W
*
ERROR at line 1:
ORA-28003: password verification for the specified password
failed
ORA-20025: Password must contain at least 2 digit(s)

SQL> ALTER USER tom IDENTIFIED BY Strong_pass_65W;

User altered.

SQL>
```

Notice that TOM falls under the rules of the password checking function defined in the DEFAULT profile even if being granted the SYSDBA privilege.

5. Set the password verification function to NULL in the DEFAULT profile. In a production environment, the password verification function should be set to a password verification function in the DEFAULT profile. You use simple passwords in the course for ease of remembrance.

```
SQL> CONNECT / AS SYSDBA
```

```
Connected.
SQL> ALTER PROFILE default LIMIT
           PASSWORD_LIFE_TIME unlimited
           FAILED_LOGIN_ATTEMPTS unlimited
           PASSWORD_VERIFY_FUNCTION null;
2    3    4
Profile altered.

SQL>
```

6. Reset the password of `TOM` to its initial value and revoke the `SYSDBA`.

```
SQL> ALTER USER tom IDENTIFIED BY oracle_4U;

User altered.

SQL> REVOKE sysdba FROM tom;

Revoke succeeded.

SQL> EXIT
$
```

7. The security officer will now define different `DEFAULT` profiles within `pdb1_1` and `pdb1_2` setting the following password limits:
   - In `pdb1_1:` A life time period set to 1 minute (for the practice purpose) and no password verify function
   - In `pdb1_2:` Account locked after 2 failed login attempts only and the password verify function set to `ora12c_strong_verify_function`

   a. Set the `ORACLE_SID` and `ORACLE_HOME` to point to the CDB instance.

```
$ . oraenv
ORACLE_SID = [orcl] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
SQL>
```

   b. The PDBs are not opened. You can either open them all each time the instance is restarted as follows:

```
SQL> alter pluggable database all open;
```

```
Pluggable database altered.

SQL>
```

or create the following trigger that will open them at each instance startup. You can use the following trigger code:

```
CREATE TRIGGER open_all_PDBs
  AFTER STARTUP ON DATABASE
begin
    execute immediate 'alter pluggable database all open';
 end open_all_PDBs;
/
```

```
SQL> CREATE TRIGGER Open_All_PDBs
  after startup on database
  begin
  execute immediate 'alter pluggable database ALL open';
  end Open_All_PDBs;
/
  2    3    4    5    6
Trigger created.

SQL>
```

c. Connect to `pdb1_1` as `SYSTEM` to alter the `DEFAULT` profile.

```
SQL> CONNECT system@pdb1_1
Enter password: ******
Connected.
SQL> ALTER PROFILE default LIMIT
           PASSWORD_LIFE_TIME 1/1440
           PASSWORD_VERIFY_FUNCTION null;
  2    3
Profile altered.

SQL> COL profile format A7
SQL> COL resource_name format A32
SQL> COL limit format A30
SQL> @$HOME/labs/default_profile.sql

PROFILE RESOURCE_NAME                    LIMIT
------- ------------------------------ ------------------------
DEFAULT FAILED_LOGIN_ATTEMPTS            10
DEFAULT PASSWORD_LIFE_TIME          .0006
```

```
DEFAULT PASSWORD_REUSE_TIME            UNLIMITED
DEFAULT PASSWORD_REUSE_MAX        UNLIMITED
DEFAULT PASSWORD_VERIFY_FUNCTION       NULL
DEFAULT PASSWORD_LOCK_TIME        1
DEFAULT PASSWORD_GRACE_TIME            7

7 rows selected.

SQL>
```

d. Connect to `pdb1_2` as `SYSTEM` to alter the `DEFAULT` profile.

```
SQL> CONNECT system@pdb1_2
Enter password: ******
Connected.
SQL> ALTER PROFILE default LIMIT
   FAILED_LOGIN_ATTEMPTS 10
   PASSWORD_VERIFY_FUNCTION ora12c_strong_verify_function;

  2    3  ALTER PROFILE default LIMIT
*
ERROR at line 1:
ORA-07443: function ORA12C_STRONG_VERIFY_FUNCTION not found

SQL> CONNECT sys@pdb1_2 AS SYSDBA
Enter password: ******
Connected.
SQL> @$ORACLE_HOME/rdbms/admin/utlpwdmg.sql

Function created.


Function created.


Function created.


Grant succeeded.


Function created.


Grant succeeded.
```

```
Function created.


Grant succeeded.


Function created.


Grant succeeded.


Profile altered.

SQL> CONNECT system@pdb1_2
Enter password: ******
Connected.
SQL> ALTER PROFILE default LIMIT
   FAILED_LOGIN_ATTEMPTS 10
```

```
    PASSWORD_VERIFY_FUNCTION ora12c_strong_verify_function;
  2    3
Profile altered.

SQL> @$HOME/labs/default_profile.sql

PROFILE RESOURCE_NAME             LIMIT
------- ------------------------- -----------------------
DEFAULT FAILED_LOGIN_ATTEMPTS     10
DEFAULT PASSWORD_LIFE_TIME   180
DEFAULT PASSWORD_REUSE_TIME       UNLIMITED
DEFAULT PASSWORD_REUSE_MAX   UNLIMITED
DEFAULT PASSWORD_VERIFY_FUNCTION ORA12C_STRONG_VERIFY_FUNCTION
DEFAULT PASSWORD_LOCK_TIME   1
DEFAULT PASSWORD_GRACE_TIME       7

7 rows selected.

SQL>
```

e. Connect to the `root` container of `cdb1` as `SYSTEM` and display the `DEFAULT` profile.

```
SQL> CONNECT system
Enter password: ******
```

```
Connected.
SQL> @$HOME/labs/default_profile.sql

PROFILE RESOURCE_NAME                 LIMIT
------- ---------------------------- ------------------
DEFAULT FAILED_LOGIN_ATTEMPTS         10
DEFAULT PASSWORD_LIFE_TIME       180
DEFAULT PASSWORD_REUSE_TIME           UNLIMITED
DEFAULT PASSWORD_REUSE_MAX       UNLIMITED
DEFAULT PASSWORD_VERIFY_FUNCTION      NULL
DEFAULT PASSWORD_LOCK_TIME       1
DEFAULT PASSWORD_GRACE_TIME           7

7 rows selected.

SQL>
```

Notice that the `root` container has its own `DEFAULT` profile.

f. Set the password verification function to `NULL` in the `DEFAULT` profile. Set the password life time to `unlimited` so that passwords do not expire during the course. You use simple passwords in the course for ease of remembrance.

```
SQL> ALTER PROFILE default LIMIT
   FAILED_LOGIN_ATTEMPTS unlimited
       PASSWORD_LIFE_TIME unlimited
       PASSWORD_VERIFY_FUNCTION null;
  2    3    4
Profile altered.

SQL> CONNECT system@pdb1_2
Enter password: ******
Connected.
SQL> ALTER PROFILE default LIMIT
   FAILED_LOGIN_ATTEMPTS unlimited
       PASSWORD_LIFE_TIME unlimited
       PASSWORD_VERIFY_FUNCTION null;
  2    3    4
Profile altered.

SQL>
```

```
SQL> CONNECT system@pdb1_1
Enter password:
Connected.
SQL> ALTER PROFILE default LIMIT
   FAILED_LOGIN_ATTEMPTS unlimited
       PASSWORD_LIFE_TIME unlimited
       PASSWORD_VERIFY_FUNCTION null;
  2    3    4
Profile altered.

SQL> EXIT
$
```

Practice 4-3: Protecting the Data Dictionary

Overview

In this practice, you will verify that the data dictionary is protected from users' visibility.

Tasks

1. After creating an Oracle database, what action do you need to take to prevent users with the *ANY* privilege from using their privileges against the data dictionary? Which types of users require the *ANY* privilege?
   Verify that the `O7_DICTIONARY_ACCESSIBILITY` parameter is set to `FALSE`. This restricts access to the data dictionary to users with the `SELECT_CATALOG_ROLE` or `SELECT ANY DICTIONARY` privilege. Users who require the `*ANY*` privilege may be DBAs who need privileges to create, alter, and drop objects, perform data manipulation language (DML), and select objects in any schema. Note that in Oracle Database 12*c*, the default value for `O7_DICTIONARY_ACCESSIBILITY` is `FALSE`.
   a. Use the `oraenv` utility to set the `ORACLE_SID` environment variable to the `orcl` value.

```
$ . oraenv
ORACLE_SID = [cdb1] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

   b. Display the value for `O7_DICTIONARY_ACCESSIBILITY` parameter.

```
$ sqlplus system
Enter password: ******
SQL> SHOW PARAMETER DICTIONARY

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
O7_DICTIONARY_ACCESSIBILITY          boolean     FALSE
SQL>
```

2. Which users have been granted `SELECT_CATALOG_ROLE`?

```
SQL> COL GRANTEE FORMAT A20
SQL> COL GRANTED_ROLE FORMAT A22
SQL> SELECT * FROM dba_role_privs
     WHERE GRANTED_ROLE LIKE 'SELECT_CATALOG%';

GRANTEE              GRANTED_ROLE          ADM DEF COM
------------------- --------------------- --- --- ---
SH                   SELECT_CATALOG_ROLE   NO  YES NO
SEC                  SELECT_CATALOG_ROLE   NO  YES NO
SYS                  SELECT_CATALOG_ROLE   YES YES NO
IX                   SELECT_CATALOG_ROLE   NO  YES NO
```

```
OEM_MONITOR          SELECT_CATALOG_ROLE   NO  YES YES
SYSBACKUP            SELECT_CATALOG_ROLE   NO  YES YES
DBA                  SELECT_CATALOG_ROLE   YES YES YES
IMP_FULL_DATABASE    SELECT_CATALOG_ROLE   NO  YES YES
EXP_FULL_DATABASE    SELECT_CATALOG_ROLE   NO  YES YES
EM_EXPRESS_BASIC     SELECT_CATALOG_ROLE   NO  YES YES

10 rows selected.

SQL>
```

3. Which users have the `SELECT ANY DICTIONARY` privilege?

```
SQL> SELECT * FROM dba_sys_privs
     WHERE privilege = 'SELECT ANY DICTIONARY';

GRANTEE              PRIVILEGE                          ADM COM
-------------------- ---------------------------------- --- ---
IX                   SELECT ANY DICTIONARY              NO  NO
SYSBACKUP            SELECT ANY DICTIONARY              NO  YES
OLAPSYS              SELECT ANY DICTIONARY              NO  YES
DBA                  SELECT ANY DICTIONARY              YES YES
WMSYS                SELECT ANY DICTIONARY              NO  YES
SYSDG                SELECT ANY DICTIONARY              NO  YES
ORACLE_OCM           SELECT ANY DICTIONARY              NO  YES
OEM_MONITOR          SELECT ANY DICTIONARY              NO  YES
DBSNMP               SELECT ANY DICTIONARY              NO  YES

9 rows selected.

SQL>
```

4. Verify that `SYSTEM` cannot view the `SYS.ENC$` nor the `SYS.LINK$` tables although being granted the `SELECT ANY DICTIONARY` privilege.

```
SQL> SELECT * FROM SYS.ENC$;
```

```
SELECT * FROM SYS.ENC$
                  *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> SELECT * FROM SYS.LINK$;
SELECT * FROM SYS.LINK$
                  *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> SELECT count(*) FROM SYS.TAB$;

  COUNT(*)
----------
      2446

SQL>
```

5. Verify that `SYS` can view the `SYS.ENC$`.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT * FROM SYS.ENC$;

no rows selected

SQL> EXIT
$
```

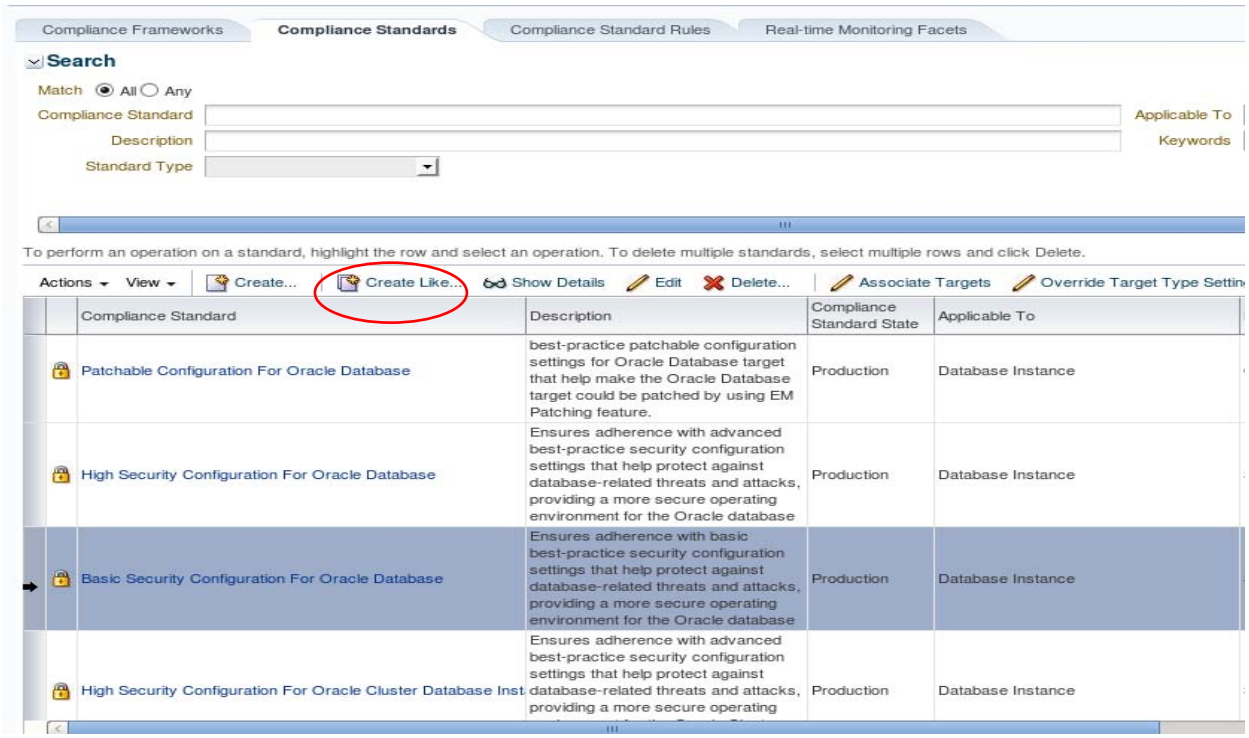
Practice 4-4: Investigating Security Violations Against Compliance Framework

Overview

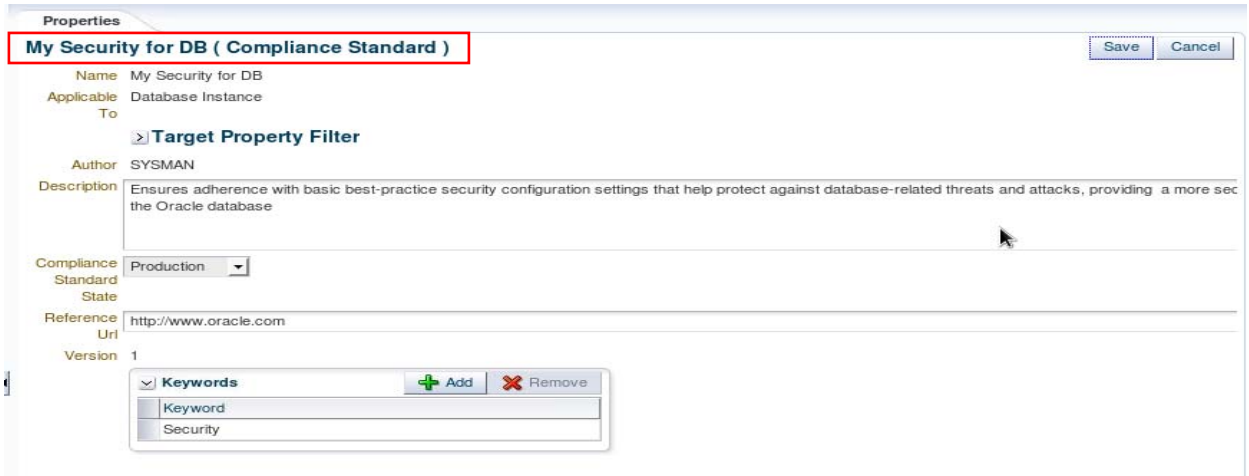
In this practice, you investigate the security violations existing in the `orcl`, `cdb1`, `pdb1_1` and `pdb1_2` databases against the predefined compliance standard, called **Basic Security Configuration For Oracle Database**. Assign both of your database instances to this compliance standard. Then view the compliance evaluation results.

Tasks

1. To assign compliance standards to your database instances, navigate to **Enterprise > Compliance > Library**.
2. Click the **Compliance Standards** tabbed page, and then the ">" icon before Search.
3. Select **Database Instance** from the Applicable To drop-down and click **Search**.
4. Because you want to ensure that there are no unexpected changes coming from predefined standards (which may be updated in the future), you create your own set. Select **Basic Security Configuration For Oracle Database** and click **Create Like**.

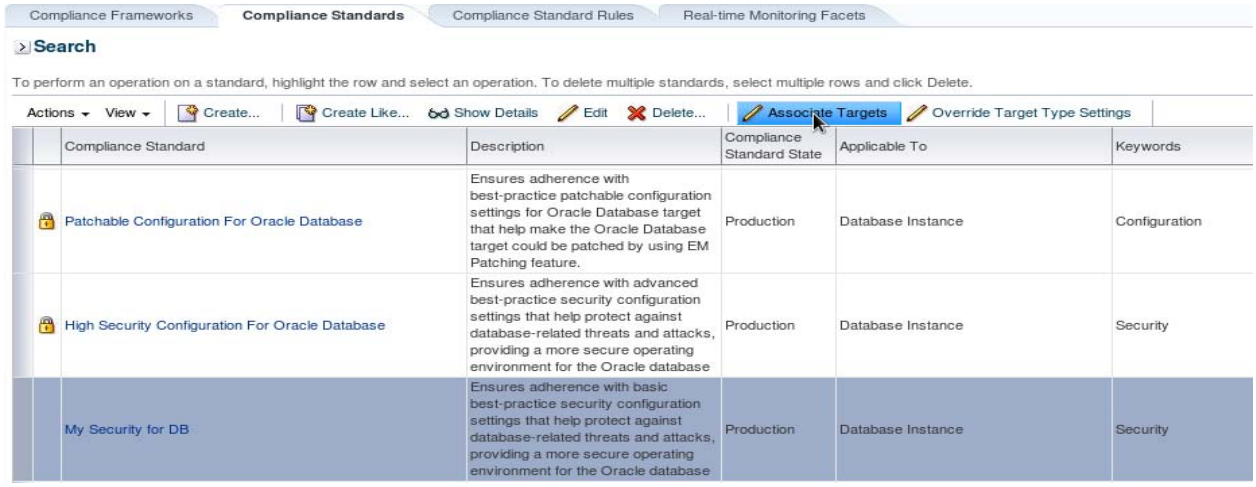


5. Enter **My Security for DB** as Name and click **Continue**.

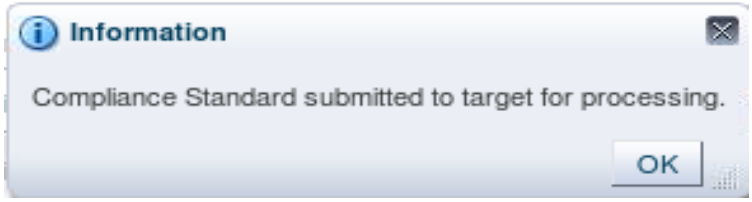


6. Click **Save**. Then **OK**.

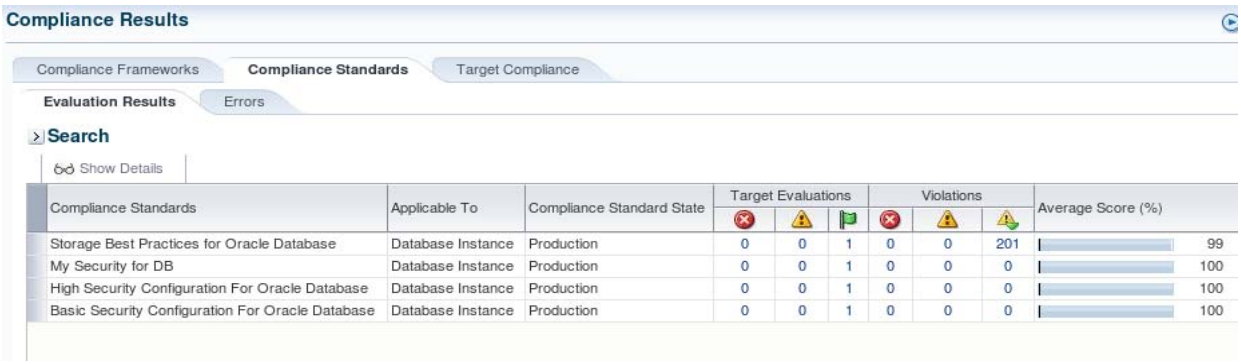
7. Select **My Security for DB** and click **Associate Targets**.



8. To associate targets, click **Add**.
9. On the "Search and Select: Targets" window, select the `orcl` database instance and click the **Select** button.
10. Click **OK**.
11. Read the Save Association message and click **Yes**.
12. You should receive the information that the compliance standard is submitted for processing. Click **OK**.



13. Repeat the previous two steps to associate the **`cdb1`** database instance to this compliance standard if you wish to investigate security violations in this target instance. To retrieve the `cdb1` instance, add the `cdb1` instance as a possible target managed in EM Cloud Control. (Execute the steps described in practice 3-2 step 9)
14. To evaluate the compliance standards, navigate to **Enterprise > Compliance > Results**.
15. Question: What is the compliance score for security best practices in each database? Click a digit under Target Evaluations in the **Compliance Standards** tab. If there is no result in the **Compliance Standards** tab, click the **Target Compliance** tab and click a digit under Evaluations.
    You may get different results than those displayed below.

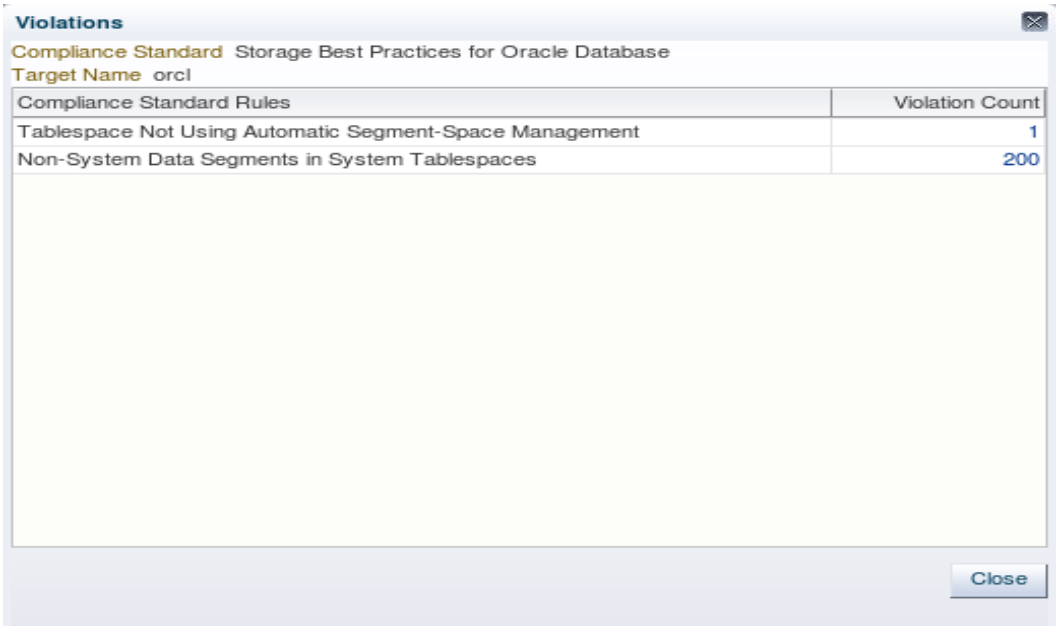


Compliance Standards	Applicable To	Compliance Standard State	Target Evaluations			Violations			Average Score (%)
Storage Best Practices for Oracle Database	Database Instance	Production	0	0	1	0	0	201	99
My Security for DB	Database Instance	Production	0	0	1	0	0	0	100
High Security Configuration For Oracle Database	Database Instance	Production	0	0	1	0	0	0	100
Basic Security Configuration For Oracle Database	Database Instance	Production	0	0	1	0	0	0	100

16. Possible answer: In this example it is 100%. Close the Compliant Targets if you were looking at Target Evaluations in the **Compliance Standards** tab or Compliant Standards page if you were looking at Evaluations in the **Target Compliance** tab.

Target Name	Last Evaluation Date	Compliance Score (%)
orcl	4/23/2013	100

17. In the **Target Compliance** tab, click the Violation link, then the Violation Count link: The database does not conform to the compliance standard rules as recommended by Oracle Corporation. You may get different results than those displayed below.



18. Close the “Violations” window.
19. Log out Enterprise Manager Cloud Control by clicking the **Logout** button.

Practices for Lesson 5: Securing Network Services

Chapter 5

Practices for Lesson 5: Overview

Practices Overview

In these practices, you will implement the network security features like configuring the listener on another port, securing the listener administration, and creating ACLs to restrict access by users to network services.

Practice 5-1: Configuring the Listener on Another Port

Overview

In this practice, you create a listener on an alternate port.

Tasks

1. Configure the listener to use an alternate port. Your network configuration files are stored in the `$TNS_ADMIN` directory (`/home/oracle/labs/NET`). Then, start your listener.
   a. Create the `/home/oracle/labs/NET` directory.

```
$ mkdir /home/oracle/labs/NET
$
```

   b. Set the `TNS_ADMIN` environment variable to `/home/oracle/labs/NET` directory.

```
$ export TNS_ADMIN=/home/oracle/labs/NET
$
```

   c. Use Oracle Net Manager to create a `listener.ora` file for a separate listener.

```
$ netmgr
```

Step	Page	Action
a.	Oracle Net Manager - /home/oracle/labs/NET	Expand **Local**. Select **Listeners**. Click **Create** (green "+" icon).
b.	Choose Listener Name	Enter **LISTEN1** as the listener name. Click OK.
c.	Oracle Net Manager	Click **Add Address**.
d.	Address1 tab	Enter the following information: Port: **13001** Verify the host name. Host : *<Your hostname>* Change Listening Locations to **General Parameters**.
e.	General tab	Click the **Logging & Tracing** tab. Deselect "**Enable ADR**." Enter the following in the Log File field: **/home/oracle/labs/NET/listen1.log**
f.	General tab	Select **File > Save As**. Find the **/home/oracle/labs/NET** Directory. Click **OK**. Click **Exit**.

2. Start the `LISTEN1` listener with the `lsnrctl` utility. Note where the log file is located.

```
$ lsnrctl start LISTEN1

LSNRCTL for Linux: Version 12.1.0.1.0 - Production on 14-JUN-
2013 06:49:04

Copyright (c) 1991, 2013, Oracle.  All rights reserved.

Starting /u01/app/oracle/product/12.1.0/dbhome_1/bin/tnslsnr:
please wait...

TNSLSNR for Linux: Version 12.1.0.1.0 - Production
System parameter file is /home/oracle/labs/NET/listener.ora
Log messages written to /home/oracle/labs/NET/listen1.log
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<your
hostname>)(PORT=13001)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<your
hostname>)(PORT=13001)))
STATUS of the LISTENER
------------------------
Alias                     LISTEN1
Version                   TNSLSNR for Linux: Version 12.1.0.1.0
- Production
Start Date                14-JUN-2013 06:49:04
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /home/oracle/labs/NET/listener.ora
Listener Log File         /home/oracle/labs/NET/listen1.log
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<your
hostname>)(PORT=13001)))
The listener supports no services
The command completed successfully
$
```

3. Display the new network configuration files and the first log created.

```
$ ls /home/oracle/labs/NET
listen1.log  listener.ora  sqlnet.ora
$
```

a. View the `listener.ora` file.

```
$ more /home/oracle/labs/NET/listener.ora
```

```
# listener.ora Network Configuration File:
/home/oracle/labs/NET/listener.ora
# Generated by Oracle configuration tools.

LOG_DIRECTORY_LISTEN1 = /home/oracle/labs/NET

LISTEN1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = <your hostname>)(PORT =
13001))
  )

LOG_FILE_LISTEN1 = listen1.log

DIAG_ADR_ENABLED_LISTEN1 = OFF

$
```

b. View the `sqlnet.ora` file.

```
$ more /home/oracle/labs/NET/sqlnet.ora
# sqlnet.ora Network Configuration File:
/home/oracle/labs/NET/sqlnet.ora
# Generated by Oracle configuration tools.

ADR_BASE = /u01/app/oracle

$
```

c. View the `listen1.log` file.

```
$ more /home/oracle/labs/NET/listen1.log

TNSLSNR for Linux: Version 12.1.0.1.0 - Production on 14-JUN-
2013 06:49:04

Copyright (c) 1991, 2013, Oracle.  All rights reserved.

System parameter file is /home/oracle/labs/NET/listener.ora
Log messages written to /home/oracle/labs/NET/listen1.log
Trace information written to
/u01/app/oracle/product/12.1.0/dbhome_1/network/trace/listen1.tr
c
Trace level is currently 0

Started with pid=29243
```

```
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<your
hostname>)(PORT=13001)))
Listener completed notification to CRS on start

TIMESTAMP * CONNECT DATA [* PROTOCOL INFO] * EVENT [* SID] *
RETURN CODE
WARNING: Subscription for node down event still pending
14-JUN-2013 06:49:04 * (CONNECT_DATA=(CID=(PROGRAM=)(HOST=<your
hostname>)(USER=oracle))(COMMAND=status
)(ARGUMENTS=64)(SERVICE=LISTEN1)(VERSION=202375424)) * status *
0
$
```

4. Create a net service name to allow connections to your `ORCL` service. Your service name is `O1`. Use the Net Manager tool to create this entry.
   a. Set the `LOCAL_LISTENER` parameter in the `ORCL` instance to the new address list of Oracle Net local listener that is, listeners that run on the same system as this instance. By default when the parameter is set to no value, the `PORT` is by default `1521`.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> show parameter local_listener

NAME                                 TYPE        VALUE
------------------------------ ----------- --------------
local_listener                        string

SQL> ALTER SYSTEM SET local_listener =
     '(ADDRESS = (PROTOCOL=TCP)(HOST=localhost)(PORT=13001))'
     SCOPE=BOTH;
  2  3
System altered.

SQL> EXIT
$
```

   b. Create a net service name. Invoke `NETMGR`.

```
$ netmgr
```

Step	Page	Action
a.	Oracle Net Manager - `/home/oracle/labs/NET`	Expand **Local**. Select **Service Naming**. Click **Create** (green "+" icon).
b.	Net Service Name Wizard: Welcome	Enter **O1** as Net Service Name. Click **Next**.
c.	Net Service Name Wizard, page 2 of 5: Protocol	Select **TCP/IP** (Internet Protocol). Click **Next**.
d.	Net Service Name Wizard, page 3 of 5: Protocol Settings	Enter the following information: Host : <*Your hostname*> Port: **13001** Click **Next**.
e.	Net Service Name Wizard, page 4 of 5: Service	Enter the following information: Service Name: **`orcl`** Click **Next**.
f.	Net Service Name Wizard, page 5 of 5: Test	Click **Test**.
g.	Connection Test	Message indicates a failure. Click **Change Login**.
h.	Change Login	Enter the following information: Username: **`SYSTEM`** Password: **`oracle_4U`** Click **OK**.
i.	Connection Test	Click **Test**. Expect connecting to the database to take a few seconds to complete. Repeat until it succeeds. Message: `Connection test was successful` Click **Close**.
j.	Net Service Name Wizard, page 5 of 5: Test	Click **Finish**.
k.	Oracle Net Manager - - `/home/oracle/labs/NET`	From the menu, select File > **Save Network Configuration**. Select File > **Exit**.

5. Verify that the net service name that you created is working. Connect from your student computer to the service on the instructor's PC by using the net service name. Start SQL*Plus on the student PC, and then connect by using the net service name.

```
$ sqlplus system@O1

Enter password:
```

```
Last Successful login time: Fri Jun 14 2013 07:44:44 +00:00

Connected to: ******
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> select name from v$database;

NAME
---------
ORCL

SQL> EXIT
$
```

Practice 5-2: Securing the Listener Administration

Overview

In this practice, you use the listener that you configured in the previous practice.

The environment variable `$TNS_ADMIN` points to the directory where the network administration files are located. This variable was set in task 1 of the previous practice.

Tasks

1. Prevent online administration of the listener and test the setting by performing the following steps:
   a. Set up the listener to prevent online administration. Do not forget to include your listener name. Add the line `ADMIN_RESTRICTIONS_LISTEN1=ON` to the `listener.ora` file. Edit the `listener.ora` file on the server with your favorite editor; `gedit` is suggested.

```
$ cd $TNS_ADMIN
$ gedit listener.ora
Add ADMIN_RESTRICTIONS_LISTEN1=ON

cat of your file should look like this:
$ cat listener.ora
# listener.ora Network Configuration File:
/home/oracle/labs/NET/listener.ora
# Generated by Oracle configuration tools.

LOG_DIRECTORY_LISTEN1 = /home/oracle/labs/NET

LISTEN1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = <Your hostname>)(PORT =
13001))
  )

ADMIN_RESTRICTIONS_LISTEN1=ON
LOG_FILE_LISTEN1 = LISTEN1.log

DIAG_ADR_ENABLED_LISTEN1 = OFF
$
```

   b. Stop and start your listener to force the `listener.ora` file to be read.

```
$ lsnrctl

LSNRCTL> SET CURRENT_LISTENER listen1
Current Listener is listen1
LSNRCTL> stop
```

```
Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=<Your
hostname>)(PORT=13001))
The command completed successfully stop LISTEN1

LSNRCTL> start

Starting /u01/app/oracle/product/12.1.0/dbhome_1/bin/tnslsnr:
please wait...

TNSLSNR for Linux: Version 12.1.0.1.0 - Production
System parameter file is /home/oracle/labs/NET/listener.ora
Log messages written to /home/oracle/labs/NET/listen1.log
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<Your
hostname>)(PORT=13001)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<Your
hostname>)(PORT=13001)))
STATUS of the LISTENER
------------------------
Alias                     LISTEN1
Version                   TNSLSNR for Linux: Version 12.1.0.1.0
- Production
Start Date                14-JUN-2013 07:54:37
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /home/oracle/labs/NET/listener.ora
Listener Log File         /home/oracle/labs/NET/listen1.log
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<Your
hostname>)(PORT=13001)))
The listener supports no services
The command completed successfully
LSNRCTL>
```

c. Attempt online administration. Set the trace level by using the following command:

```
LSNRCTL> SET TRC_LEVEL user
```

This verifies that you cannot administer the listener online.

```
LSNRCTL> SET TRC_LEVEL user
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<Your
hostname>)(PORT=13001)))
TNS-12508: TNS:listener could not resolve the COMMAND given
LSNRCTL> exit
```

```
$
```

2. Edit the `listener.ora` file, removing the online administration restriction by deleting the `ADMIN_RESTRICTIONS_LISTEN1=ON` entry.

```
$ gedit listener.ora
Remove ADMIN_RESTRICTIONS_LISTEN1=ON
```

3. Reload the `listener.ora` file. Do not forget to set your current listener.

```
$ lsnrctl

LSNRCTL> SET CURRENT_LISTENER listen1
Current Listener is listen1
LSNRCTL> reload
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<Your
hostname>)(PORT=13001)))
The command completed successfully
LSNRCTL>
```

4. Test the change. In Listener Control, set the trace level by using the following command:

   `LSNRCTL> SET TRC_LEVEL user`

   This verifies that you can currently administer the listener online.

```
LSNRCTL> SET TRC_LEVEL user
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=EDRSR1P1)(PORT=13001))
)
listen1 parameter "trc_level" set to user
The command completed successfully
LSNRCTL> exit
$
```

Practice 5-3: Configure the Listener to Allow Access Only from Your Client Computer *(optional)*

Overview

In this practice, you configure your listener on the server to allow access only from your client computer.

Tasks

1. Determine the IP address of your neighbor's PC. Ask your neighbor to use `nslookup` `hostname` to determine the IP address of his/her computer. This command uses the grave (`) punctuation marks to execute the `hostname` command. IP address: _____

```
$ nslookup `hostname`

Server:   192.0.2.1
Address:  192.0.2.1#53

Name:     His/Her_servername
Address:  192.0.2.254
$
```

2. Set up Oracle Net Services to allow connections from his/her client computer and deny all others. When `tcp.invited_nodes` is set, all nodes except those invited are excluded. The `tcp.invited_nodes` and `tcp.excluded_nodes` parameters can be used independently; if `tcp.excluded_nodes` is used by itself, only the nodes listed are blocked. If `tcp.invited_nodes` is used by itself, only `tcp.invited_nodes` are allowed to connect. If both are used together, the `tcp.invited_nodes` list takes precedence.
   a. Stop the listener before applying changes to the `sqlnet.ora` file.

```
$ lsnrctl

LSNRCTL> set current_listener listen1
Current Listener is listen1
LSNRCTL> stop
Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=<Your
hostname>)(PORT=13001))
The command completed successfully
```

```
LSNRCTL> exit
$
```

   b. Use `gedit` to edit the `sqlnet.ora` file. Include his/her server host name in `tcp.invited_nodes`. Add the lines shown in bold in the following code. Substitute his/her IP address and add your own host server name.

```
# sqlnet.ora Network Configuration File:
# Generated by Oracle configuration tools.
```

```
ADR_BASE = /u01/app/oracle
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)

tcp.validnode_checking = YES
tcp.invited_nodes = (<your hostname>, <neighbor's hostname>)
```

```
$ cd $TNS_ADMIN
$ gedit sqlnet.ora
```

c. Start your listener for these changes to be applied to the listener.

```
$ lsnrctl

LSNRCTL> set current_listener listen1
Current Listener is listen1
LSNRCTL> start
```

```
Starting /u01/app/oracle/product/12.1.0/dbhome_1/bin/tnslsnr:
please wait...

TNSLSNR for Linux: Version 12.1.0.1.0 - Production
System parameter file is /home/oracle/labs/NET/listener.ora
Log messages written to /home/oracle/labs/NET/listen1.log
Listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=edRSr1p1.us.oracle.com
)(PORT=13001)))

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=EDRSR1P1)(PORT=13001))
)
STATUS of the LISTENER
------------------------
Alias                     listen1
Version                   TNSLSNR for Linux: Version 12.1.0.1.0
- Production
Start Date                14-JUN-2013 09:15:18
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /home/oracle/labs/NET/listener.ora
Listener Log File         /home/oracle/labs/NET/listen1.log
Listening Endpoints Summary...
```

```
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<Your
hostname>)(PORT=13001)))
The listener supports no services
The command completed successfully
LSNRCTL>
LSNRCTL> exit
$
```

3. Ask your neighbor to test by attempting to connect to your Oracle server instance. He will use the `EZCONNECT` connect string that does not require a service name to be in the `tnsnames.ora` file. The backslash is required to escape the quote.

```
$ sqlplus system@\'<your hostname>:13001/orcl\'
Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics,
Real Application Testing and Unified Auditing options

SQL> EXIT
$
```

4. Ask another student, whose PC's address is not one of the invited nodes, to use the `EZCONNECT` style connection string and attempt to connect to your listener.

```
$ sqlplus system@\<your hostname>:13001/orcl\'

SQL*Plus: Release 12.1.0.1.0 Production on Tue Sep 10 09:26:15
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter password:
ERROR:
ORA-12547: TNS:lost contact



Enter user-name:
$
```

5. Restore the listener so that it accepts any connections by removing the two parameters or by just removing the `sqlnet.ora` file.

```
$ cd $TNS_ADMIN
$ rm sqlnet.ora
$ lsnrctl stop listen1
```

```
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<Your
hostname>)(PORT=13001)))
```

```
The command completed successfully
$
```

6. Analyze the listener log file. Find the following entries: `status`, `stop`, a failed attempt at online administration, and a rejected connection.

```
$ cd /home/oracle/labs/NET
```

```
$ less listen1.log
```

```
26-JUN-2013 01:57:56 *
(CONNECT_DATA=(CID=(PROGRAM=)(HOST=EDRSR32P1)(USER=oracle))(COMM
AND=status)(ARGUMENTS=64)(SERVICE=listen1)(VERSION=202375424)) *
status * 0
…
26-JUN-2013 01:58:14 * trc_level * 12508
TNS-12508: TNS:listener could not resolve the COMMAND given
…
26-JUN-2013 01:57:56 *
(CONNECT_DATA=(CID=(PROGRAM=)(HOST=your_server)(USER=oracl
e))(COMMAND=status)(ARGUMENTS=64)(SERVICE=listen1)(VERSION=20237
5424)) * status     * 0
…
26-JUN-2013 02:27:56 * (CONNECT_DATA=(CID=(PROGRAM=)(HOST=
your_server)(USER=oracl
e))(COMMAND=stop)(ARGUMENTS=64)(SERVICE=listen1)(VERSION=2023754
24)(CRS=ON)) *     stop * 0
…
```

7. Clean up the listener configuration.
   a. Set the `TNS_ADMIN` environment variable to `$ORACLE_HOME/network/admin`.

```
$ export TNS_ADMIN=$ORACLE_HOME/network/admin
$
```

   b. Reset the `LOCAL_LISTENER` parameter to the default value.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> show parameter local_listener

NAME             TYPE    VALUE
---------------- ------ -------------------------
```

```
local_listener  string (ADDRESS = (PROTOCOL=TCP)(HOST
                        =localhost)(PORT=13001))

SQL> ALTER SYSTEM RESET local_listener;

System altered.

SQL>
```

c. Restart the instance.

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289400 bytes
Variable Size             293601544 bytes
Database Buffers          197132288 bytes
Redo Buffers                8036352 bytes
Database mounted.
Database opened.
SQL> show parameter local_listener

NAME                    TYPE     VALUE
----------------------- ----------- --------------
local_listener          string
SQL> EXIT
$
```

d. Remove all network files created for the purpose of these practices 5.

```
$ rm /home/oracle/labs/NET/*
$
```

e. Verify the status of the `LISTENER` listener.

```
$ lsnrctl status

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
```

```
Version                   TNSLSNR for Linux: Version 12.1.0.1.0
- Production
Start Date                12-JUN-2013 00:45:52
Uptime                    5 days 2 hr. 17 min. 16 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/app/oracle/product/12.1.0/dbhome_1/network/admin/listener.o
ra
Listener Log File
/u01/app/oracle/diag/tnslsnr/EDRSR32P1/listener/alert/log.xml
Listening Endpoints Summary...
    (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<Your
hostname>)(PORT=1521)))
Services Summary...
Service "cdb1" has 1 instance(s).
  Instance "cdb1", status READY, has 1 handler(s) for this
service...
Service "cdb1XDB" has 1 instance(s).
  Instance "cdb1", status READY, has 1 handler(s) for this
service...
Service "em12rep" has 1 instance(s).
  Instance "em12rep", status READY, has 1 handler(s) for this
service...
Service "em12repXDB" has 1 instance(s).
  Instance "em12rep", status READY, has 1 handler(s) for this
service...
Service "orcl" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
Service "orclXDB" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
Service "pdb1_1" has 1 instance(s).
  Instance "cdb1", status READY, has 1 handler(s) for this
service...
Service "pdb1_2" has 1 instance(s).
  Instance "cdb1", status READY, has 1 handler(s) for this
service...
The command completed successfully
$
```

Practices for Lesson 6: Implementing Basic and Strong Authentication

Chapter 6

Practices for Lesson 6: Overview

Practices Overview

In these practices, you will implement the basic password and OS authentication, secure the passwords, restrict database links and manage authentication of common and local users in CDBs and PDBs.

Practice 6-1: Using Basic OS Authentication Method

Overview

In this practice, you will in a first step explore basic authentication techniques for implementing a no-password login and the weaknesses of this method.

Assumptions

In your company, there are several situations that require exceptions to the standard password policies. Batch jobs should not have passwords embedded in the script or command line.

Tasks

1. A batch job that runs as the `fred` operating system user should be able to connect to the database as the `FRED` database user without having to embed the database password in the batch file.

   Configure `OS_AUTHENT_PREFIX` to allow the OS user and database user to have the same string. What is the default value of `OS_AUTHENT_PREFIX`? Is `OS_AUTHENT_PREFIX` a static parameter?

   Connect to the database as the `SYS` user. Set the `OS_AUTHENT_PREFIX` parameter to ''.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> show parameter OS_AUTHENT_PREFIX

NAME                        TYPE       VALUE
--------------------------- ---------- -----
os_authent_prefix           string     ops$
SQL>
SQL> column value format A10
SQL> column name format A24
SQL> select name, value, isdefault, ISSYS_MODIFIABLE
     from v$parameter
     where name = 'os_authent_prefix';
  2    3
NAME                        VALUE      ISDEFAULT ISSYS_MOD
--------------------------- ---------- --------- ---------
```

```
os_authent_prefix         ops$       TRUE       FALSE

SQL> ALTER SYSTEM SET OS_AUTHENT_PREFIX='';
ALTER SYSTEM SET OS_AUTHENT_PREFIX=''
                 *
ERROR at line 1:
ORA-02095: specified initialization parameter cannot be modified

SQL> ALTER SYSTEM SET OS_AUTHENT_PREFIX='' SCOPE=SPFILE;

System altered.

SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289400 bytes
Variable Size             264241416 bytes
Database Buffers          226492416 bytes
Redo Buffers                8036352 bytes
Database mounted.
Database opened.
SQL>
```

2. Create the database user `FRED`, using the `IDENTIFIED EXTERNALLY` clause. Allow `FRED` to connect to the database.

   As the `SEC` user, create the `FRED` user and grant the `CREATE SESSION` privilege.

```
SQL> CONNECT SEC
Enter password: *****
Connected.
SQL>
SQL> CREATE USER FRED IDENTIFIED EXTERNALLY;

User created.

SQL>
SQL> GRANT CREATE SESSION TO FRED;
```

```
Grant succeeded.

SQL> ALTER USER FRED
     DEFAULT TABLESPACE EXAMPLE
     QUOTA UNLIMITED ON EXAMPLE;
  2    3
User altered.

SQL> EXIT
$
```

3. Test the connection as the `fred` user. Log in to the OS as the `fred` user. The OS password for `fred` is `oracle`. Connect to the database with the "`/`" connect string.

```
$ su - fred
Password: *****
$ . oraenv
ORACLE_SID = [fred] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus /

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> SHOW USER
USER is "FRED"
SQL> EXIT
$ exit
logout
$
```

Notice that any connection using an OS or password authentication provides the "Last Successful Logon Time" for non-SYS users. You can see it in the SQL*Plus banner. You will see the message when you connected at least once before.

Practice 6-2: Observing Passwords in Database Links

Overview

In this practice, you explore the protection of passwords for database links in Oracle Database 12*c*.

Tasks

1. Create and test a database link in the `PDB1_1` pluggable database. Log in as the `oracle` OS user. As the `SYSTEM` database user, create a database link for the `HR` user to the `ORCL` database.

```
CREATE PUBLIC DATABASE LINK test_hr
            CONNECT TO hr IDENTIFIED BY oracle_4U
            USING 'ORCL';
```

   Note: Only users with the `CREATE PUBLIC DATABASE LINK` privilege can execute this command.

```
$ sqlplus system@pdb1_1
Enter password: ******

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
SQL>
SQL> CREATE PUBLIC DATABASE LINK test_hr
            CONNECT TO hr IDENTIFIED BY oracle_4U
            USING 'ORCL';
2    3
Database link created.

SQL>
```

2. Test the database connection as the database user `SCOTT` by selecting from the `EMPLOYEES` table through the database link.

   Any database user will be able to use this database link because it is declared `PUBLIC`. Connected as `SYSTEM`, open the `SCOTT` account, and then test the database link.

```
SQL> ALTER USER scott IDENTIFIED BY oracle_4U ACCOUNT UNLOCK;

User altered.

SQL> connect scott@pdb1_1
Enter password: ******
Connected.
```

```
SQL> select max(salary) from employees@test_hr;

MAX(SALARY)
-----------
      24000

SQL>
```

3. View the data dictionary information about the database link. Find the username and password as they are stored in the database.
   a. Connect as `SYSTEM` and query the `DBA_DB_LINKS` view for the database link information.

```
SQL> CONNECT system@pdb1_1
Enter password: ******
Connected.
SQL> COL username FORMAT A16
SQL> COL owner FORMAT A16
SQL> COL db_link FORMAT A16
SQL> SELECT owner, db_link, username FROM DBA_DB_LINKS;

OWNER            DB_LINK          USERNAME
---------------- ---------------- ----------------
PUBLIC           TEST_HR          HR

SQL> SELECT name, authusr, authpwd, passwordx, authpwdx
     FROM   SYS.LINK$;
  2       FROM LINK$
               *
ERROR at line 2:
ORA-01031: insufficient privileges

SQL>
```

   The `SYSTEM` user is granted the `SELECT ANY DICTIONARY` privilege but cannot view the `SYS.LINK$` table.

4. View the base `SYS` table for the database links. As the `SYS` user, view the `LINK$` table. Is the password visible in this table? Describe the table to view all columns. Query the table to view passwords. Note that all passwords are encrypted. None are stored in clear text.

```
SQL> CONNECT / as sysdba
Connected.
SQL> desc link$
```

```
Name                          Null?    Type
```

```
 -------------------------- -------- -------------------
 OWNER#                     NOT NULL NUMBER
 NAME                       NOT NULL VARCHAR2(128)
 CTIME                      NOT NULL DATE
 HOST                                VARCHAR2(2000)
 USERID                              VARCHAR2(128)
 PASSWORD                            VARCHAR2(128)
 FLAG                                NUMBER
 AUTHUSR                             VARCHAR2(128)
 AUTHPWD                             VARCHAR2(128)
 PASSWORDX                            RAW(128)
 AUTHPWDX                            RAW(128)

SQL> SELECT name, authusr, authpwd, passwordx, authpwdx
     FROM LINK$;
 2
no rows selected

SQL>
```

Note that you are connected to the root container. You created the database link in the `PDB1_1` container.

```
SQL> CONNECT sys@pdb1_1 as sysdba
Enter password: ******
Connected.

SQL> SELECT name, authusr, authpwd, passwordx, authpwdx
     FROM LINK$;
 2

NAME
--------------------------------------------------------
AUTHUSR
--------------------------------------------------------
AUTHPWD
--------------------------------------------------------
PASSWORDX
--------------------------------------------------------
AUTHPWDX
--------------------------------------------------------
TEST_HR
```

```
07C3AA3161B61534381479C836FC0B4681E68548F32D28845EC40B1A
```

```
7A4A5421A6D84FE46C53B1E374BF928D0ED35AE8B1E4D9CC5E08A1F7
13471B9CB6C61ED3345FC4D8C75504AA127AD3EB564FA583EE3117BB
37209801CA3F0156C5360F0C2A14A261D6380A100F1ED93257D72C4D
ED56E34907B613BCC96C0AB90F1D9E6


SQL>
```

Practice 6-3: Restricting Database Links With Views

Overview

In this practice, you will restrict to the access to tables in the `HR` schema authorized by the `hrviewlink` database link.

Tasks

1. While you are still connected to `pdb1_1`, create the `MIKE` user and grant him the `HR_MGR` role.

```
SQL> SET ECHO ON
SQL> DROP ROLE HR_MGR;
DROP ROLE HR_MGR
          *
ERROR at line 1:
ORA-01919: role 'HR_MGR' does not exist

SQL> CREATE ROLE HR_MGR;

Role created.

SQL> DROP USER mike CASCADE;
DROP USER mike CASCADE
          *
ERROR at line 1:
ORA-01918: user 'MIKE' does not exist

SQL> CREATE USER mike identified by oracle_4U;

User created.

SQL> GRANT CREATE SESSION TO mike;

Grant succeeded.

SQL> GRANT HR_MGR to mike;

Grant succeeded.

SQL>
```

2. Create the `hrviewlink` database link.

```
SQL> CONNECT hr@pdb1_1
Enter password:
```

```
ERROR:
ORA-28000: the account is locked

Warning: You are no longer connected to ORACLE.
SQL> CONNECT system@pdb1_1
Enter password: ******
Connected.
SQL> ALTER USER hr IDENTIFIED BY oracle_4U ACCOUNT UNLOCK;

User altered.

SQL> CONNECT hr@pdb1_1
Enter password: ******
Connected.
SQL> DROP DATABASE LINK hrviewlink;
DROP DATABASE LINK hrviewlink
                   *
ERROR at line 1:
ORA-02024: database link not found

SQL> CREATE DATABASE LINK hrviewlink CONNECT TO hr IDENTIFIED BY
oracle_4U USING 'orcl';

Database link created.

SQL>
```

3. Create the `employees_vw` view and check that it allows you to retrieve `HR.EMPLOYEES@hrviewlink` rows.

```
SQL> CREATE VIEW employees_vw as
            SELECT * FROM HR.EMPLOYEES@hrviewlink;
  2
View created.

SQL> GRANT select, insert, update, delete on EMPLOYEES_VW to
HR_MGR;

Grant succeeded.

SQL> SELECT employee_id, salary
     FROM   employees@hrviewlink
     WHERE  employee_id = 206;
```

```
2    3
EMPLOYEE_ID    SALARY
----------- ----------
        206        8300

SQL>
```

4. Connect as `MIKE` and test the view.

```
SQL> CONNECT mike@pdb1_1
Enter password: ******
Connected.
SQL> UPDATE hr.EMPLOYEES_VW SET SALARY = 10000
     WHERE employee_id = 206;
  2
1 row updated.

SQL> SELECT employee_id, salary FROM hr.employees_vw
     WHERE employee_id = 206;
  2
EMPLOYEE_ID    SALARY
----------- ----------
        206       10000

SQL> ROLLBACK;

Rollback complete.

SQL>
```

5. Attempt to view some other table `HR.DEPARTMENTS` of the `HR` schema.

```
SQL> SELECT * FROM hr.departments@hrviewlink;
SELECT * FROM hr.departments@hrviewlink
                             *
ERROR at line 1:
ORA-02019: connection description for remote database not found


SQL> EXIT
$
```

Practice 6-4: Configuring the External Secure Password Store

Overview

In this practice, you will configure the External Secure Password Store to hide passwords in batch jobs scripts.

Assumptions

You successfully completed Practice 6-1 Task 1.

Tasks

The batch processes have been moved to a client machine. The batch processes will continue using the `/@netservice_name` login for database connections. However, you must follow security best practices: hence remote OS authentication (`REMOTE_OS_AUTHENT`) is not allowed. Configure the external secure password store for the `fred` user to connect as the `HR` database user.

1. Log in to the operating system as `fred`.

```
$ su - fred
Password: ******
$
```

2. Create the following directories required for this practice: `/home/fred/oracle/wallet` and `/home/fred/oracle/network`.

   Set the permissions on the wallet directory to be accessible only to `fred`.

```
$ mkdir /home/fred/oracle
$ mkdir /home/fred/oracle/wallet
$ mkdir /home/fred/oracle/network
$ ls -l /home/fred/oracle
total 8
drwxr-xr-x 2 fred users 4096 Jan 20 16:35 network
drwxr-xr-x 2 fred users 4096 Jan 20 16:35 wallet
$ chmod 700 /home/fred/oracle/wallet
$ ls -l /home/fred/oracle
total 8
drwxr-xr-x 2 fred users 4096 Jan 20 16:35 network
drwx------ 2 fred users 4096 Jan 20 16:35 wallet
$
```

3. Create and configure the client-side Oracle wallet in the following directory that is accessible only to `fred`: `/home/fred/oracle/wallet`.

   If the wallet does not exist, create the client wallet using the command `mkstore -wrl <wallet_location> -create` where <wallet_location> is the path to the directory where you want to create and store the wallet. This command creates an Oracle wallet with the auto login feature enabled at the location you specify. When auto login is enabled for a wallet, only the operating system user who created it can manage it.

   a. Use the `mkstore` utility. Set the wallet password to `welcome1`.

```
$ . oraenv
ORACLE_SID = [fred] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ mkstore -wrl /home/fred/oracle/wallet -create
Oracle Secret Store Tool : Version 11.2.0.1.0 - Production
Copyright (c) 2004, 2009, Oracle and/or its affiliates. All
rights reserved.
Enter password: ******
Enter password again: *******
$
```

b. Add credentials to the wallet using `mkstore -wrl <wallet_location> -createCredential <db_connect_string> <username> [<password>]` where <db_connect_string> is a TNS alias or any service name used to connect to the database. The service name specified in the `mkstore` command and the service name used to connect to the database (in connect /@<db_connect_string>) must be identical. Add credentials to the wallet so that `fred` can connect to the `HR` schema without a password. Set the service name to `hr_sec`, with the username `hr` and the password `oracle_4U`.

```
$ mkstore -wrl /home/fred/oracle/wallet -createCredential hr_sec
hr
Oracle Secret Store Tool : Version 11.2.0.1.0 - Production
Copyright (c) 2004, 2009, Oracle and/or its affiliates. All
rights reserved.
Your secret/Password is missing in the command line
Enter your secret/Password:     (oracle_4U)
Re-enter your secret/Password: (oracle_4U)
Enter wallet password:          (welcome1)
Create credential oracle.security.client.connect_string1
$
```

4. Still logged in as `fred`, set the `$TNS_ADMIN` environment variable to `/home/fred/oracle/network`. Edit the `.bashrc` file with `vi` or `gedit`. The `.bashrc` file is in `the /home/fred` directory. Change the `.bashrc` file by adding the following line: `export TNS_ADMIN=/home/fred/oracle/network`

   a. Change the `.bashrc` file.

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

export TNS_ADMIN=/home/fred/oracle/network
```

b. Force the changes to take effect and verify that they have.

```
$ source ./.bashrc
$ echo $TNS_ADMIN
/home/fred/oracle/network
$
```

5. Copy the `sqlnet.ora` file from `/home/oracle/labs/admin` to `/home/fred/oracle/network`.

```
$ cd /home/fred/oracle/network
$ cp /home/oracle/labs/admin/sqlnet.ora ./
$
```

6. View the `sqlnet.ora` file, and verify that the following lines are included:

```
WALLET_LOCATION =
 (SOURCE =
   (METHOD = FILE)
   (METHOD_DATA =
     (DIRECTORY =
       /home/fred/oracle/wallet)))

SQLNET.WALLET_OVERRIDE = TRUE
```

The `sqlnet.ora` file has three parameters for configuring the secure external password store: `WALLET_LOCATION`, `SQLNET.WALLET_OVERRIDE`, and `SQLNET.AUTHENTICATION.SERVICES`.

- `WALLET_LOCATION` points to the directory where the wallet resides; this parameter exists in earlier versions.
- Set the `SQLNET.WALLET_OVERRIDE` parameter to `TRUE`. This setting causes all `CONNECT /@db_connect_string` statements to use the information in the wallet at the specified location to authenticate to databases.
- If an application uses SSL for encryption, the `sqlnet.ora` parameter, `SQLNET.AUTHENTICATION_SERVICES`, specifies SSL and an SSL wallet is created. If this application wants to use secret store credentials to authenticate to databases (instead of the SSL certificate), those credentials must be stored in the SSL wallet. If `SQLNET.WALLET_OVERRIDE = TRUE`, the usernames and passwords from the wallet are used to authenticate to databases. If `SQLNET.WALLET_OVERRIDE = FALSE`, the SSL certificate is used.

```
$ cat sqlnet.ora

NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)

WALLET_LOCATION =
 (SOURCE =
   (METHOD = FILE)
```

```
    (METHOD_DATA =
      (DIRECTORY =
        /home/fred/oracle/wallet)))

SQLNET.WALLET_OVERRIDE = TRUE
```

7. Copy the `/home/oracle/labs/admin/tnsnames.ora` file to `/home/fred/oracle/network/tnsnames`.

```
$ cp /home/oracle/labs/admin/tnsnames.ora tnsnames.ora
```

8. Edit the `/home/fred/oracle/network/tnsnames.ora` file. Replace the `ORCL` alias by the `HR_SEC` alias at the beginning of the file:

```
HR_SEC =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
  )
  (CONNECT_DATA =
    (SERVICE_NAME = orcl)
  )
 )
```

9. Test the configuration by attempting to connect to the database instance with the connect string `/@hr_sec`.

```
$ sqlplus /@hr_sec

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> show user
USER is "HR"
SQL> exit
$
```

10. List the contents of the wallet. Use the `mkstore` command with the `listCredential` option. Use the following command:

```
mkstore –wrl /home/fred/oracle/wallet –listCredential
```

```
$ mkstore -wrl /home/fred/oracle/wallet -listCredential
Oracle Secret Store Tool : Version 11.2.0.1.0 - Production
Copyright (c) 2004, 2009, Oracle and/or its affiliates. All
rights reserved.
```

```
Enter wallet password:

List credential (index: connect_string username)
1: hr_sec  hr
$ exit
logout
$
```

11. As the `oracle` user, attempt to use the wallet belonging to `fred` to connect with the connect string `/@hr_sec`.
    a. Set `TNS_ADMIN` to `/home/oracle/labs/admin`. The `sqlnet.ora` file is set up to use the wallet at `/home/fred/oracle/wallet`.

```
$ export TNS_ADMIN=/home/oracle/labs/admin
$ cd $TNS_ADMIN
$
```

    b. Open the `tnsnames.ora` file from `/home/oracle/labs/admin` and edit the same way as in step 8.

```
HR_SEC =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
  )
  (CONNECT_DATA =
      (SERVICE_NAME = orcl)
   )
  )
```

```
$ gedit tnsnames.ora

$
```

    c. Test the `HR_SEC` net service name.

```
$ tnsping HR_SEC

Copyright (c) 1997, 2013, Oracle.  All rights reserved.

Used parameter files:
/home/oracle/labs/admin/sqlnet.ora



Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (ADDRESS =
(PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))) (CONNECT_DATA
= (SERVICE_NAME = orcl)))
OK (30 msec)
```

```
$
```

d. Attempt to connect using the `HR_SEC` service name with a password. Use `system`.

```
$ sqlplus /nolog

SQL*Plus: Release 12.1.0.1.0 Production on Mon Jun 17 05:35:29
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

SQL> connect system@HR_SEC
Enter password: ******
Connected
SQL> exit
$
```

e. Attempt to connect using the `HR_SEC` service name without a password. This fails because the wallet is owned by `fred` and has the restrictive permissions `rwx------` as shown in step 2.

```
$ sqlplus /nolog

SQL*Plus: Release 12.1.0.1.0 Production on Mon Jun 17 05:36:28
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

SQL> connect /@HR_SEC
ERROR:
ORA-12578: TNS:wallet open failed

SQL> exit
$
```

f. Clear the `TNS_ADMIN` environment variable.

```
$ unset TNS_ADMIN
$
```

12. To clean up after this practice, reset the `OS_AUTHENT_PREFIX` parameter to the default values in the `ORCL` instance.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
```

```
SQL> ALTER SYSTEM SET OS_AUTHENT_PREFIX='ops$' SCOPE=SPFILE;

System altered.

SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289400 bytes
Variable Size             293601544 bytes
Database Buffers          197132288 bytes
Redo Buffers                8036352 bytes
Database mounted.
```

```
Database opened.
SQL> EXIT
$
```

Practice 6-5: Connecting to a CDB or a PDB

Overview

In this practice, you will create a common user in the CDB and observe that the common user will connect with the same password in all PDBs in the CDB. In a second step, you will create a local user in each of the two PDBs of the CDB and observe how the local users connect to the PDBs.

Tasks

1. Create the common user `C##U1` in `cdb1`.

```
$ . oraenv
ORACLE_SID = [cdb1] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus system
Enter password: ******
Last Successful login time: Mon Jun 17 2013 02:46:48 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL> CREATE USER c##u1 IDENTIFIED BY oracle_4U CONTAINER=ALL;

User created.

SQL> GRANT create session TO c##u1 CONTAINER=ALL;

Grant succeeded.

SQL>
```

2. Connect as `C##U1` in the `root`.

```
SQL> CONNECT c##u1
Enter password: ******
Connected.
SQL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
SQL>
```

3. Connect as `C##U1` in `pdb1_1`.

```
SQL> CONNECT c##u1@pdb1_1
Enter password: ******
Connected.
SQL> SHOW CON_NAME

CON_NAME
------------------------------
PDB1_1
SQL>
```

4. Connect as `C##U1` in `pdb1_2`.

```
SQL> CONNECT c##u1@pdb1_2
Enter password: ******
Connected.
SQL> SHOW CON_NAME

CON_NAME
------------------------------
PDB1_2
SQL>

SQL>
```

Notice that the same password is used to connect to any container of `cdb1`.

5. Create the local user `LOCAL_EMPLOYEE` in `pdb1_1`.
   a. Connect as `SYSTEM` in `pdb1_1`.

```
SQL> CONNECT system@pdb1_1

Enter password: ******
Last Successful login time: Mon Jun 17 2013 03:13:35 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL>
```

   b. Create the local user `LOCAL_EMPLOYEE`.

```
SQL> CREATE USER local_employee IDENTIFIED BY pass_pdb1;

User created.
```

```
SQL> GRANT create session TO local_employee;

Grant succeeded.

SQL>
```

c. Connect as LOCAL_EMPLOYEE in pdb1_1.

```
SQL> CONNECT local_employee@pdb1_1
Enter password: ******
Connected.
SQL>
```

d. Connect as LOCAL_EMPLOYEE in pdb1_2.

```
SQL> CONNECT local_employee@pdb1_2
Enter password: ******
ERROR:
ORA-01017: invalid username/password; logon denied


Warning: You are no longer connected to ORACLE.
SQL>
```

6. Create the local user LOCAL_EMPLOYEE in pdb1_2.

a. Connect as SYSTEM in pdb1_2.

```
SQL> CONNECT system@pdb1_2
Enter password: ******
Connected.
SQL>
```

b. Create the local user LOCAL_EMPLOYEE.

```
SQL> CREATE USER local_employee IDENTIFIED BY pass_pdb2;

User created.

SQL> GRANT create session TO local_employee;

Grant succeeded.

SQL>
```

c. Connect as LOCAL_EMPLOYEE in pdb1_2.

```
SQL> CONNECT local_employee@pdb1_2
Enter password: ******
Connected.
SQL>
```

d. Connect as `LOCAL_EMPLOYEE` in `pdb1_1` with the password assigned to `LOCAL_EMPLOYEE` in `pdb1_2`.

```
SQL> CONNECT local_employee@pdb1_1
ERROR:
ORA-01017: invalid username/password; logon denied


Warning: You are no longer connected to ORACLE.
SQL>
SQL> EXIT
$
```

Notice that the password used by the local user to connect to `pdb1_1` and `pdb1_2` are different.

Practices for Lesson 07: Using Enterprise User Security

Chapter 7

Practices for Lesson 7: Overview

Practices Overview

In the demonstration for this lesson, you will use the Enterprise User Security to connect to a database with unknown database users, but with directory entry users.

Practice 7-1: Using Enterprise User Security

Overview

In this practice, you use a browser to execute the “Managing_Users_and_Roles_With_EUS” demonstration. The demonstration explains how to:

- Configure and register a database with an LDAP directory.
- Create and map global private schemas and global shared schemas with directory entries.
- Test the connections as unknown database users.
- Create global roles and enterprise roles, and map them together to assign enterprise roles to directory entry users.
- Test the connections of unknown database users being granted enterprise roles.
- View audited connections for unknown users.

Tasks

1. Launch a browser and enter: file:////home/oracle/labs/EUS/Managing_Users_and_Roles_with_EUS/Managing_Users_and_Roles_With_EUS.html

Practices for Lesson 8: Using Proxy Authentication

Chapter 8

Practice 8-1: Using Proxy Authentication

In this practice, you use the OCI programs that simulate an in-house developed application server: `proxy_user` and `proxy_role`. For both, the program starts by connecting to the `ORCL` database as the `HRAPP` user and creating a connection pool with 10 connections, and then it attempts to create sessions for the `PFAY` user. The conditions will vary and sometimes the sessions will fail to be created.

Task

1. If you did not create the `SEC` user in Practice 4-1, run the `/home/oracle/labs/USERS/create_sec.sh` script to create this user. As the `SEC` user, create a user to simulate a middle-tier user.
   a. Create a user with the following properties:
      Username: `HRAPP`
      Password: `HRAPP`
      (**Note:** This password is case-sensitive; it must be in uppercase.)
      `CREATE SESSION` privilege

```
$ . oraenv
ORACLE_SID = [cdb1] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus sec
Enter password: ******
Last Successful login time: Mon Jun 17 2013 03:07:45 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> CREATE USER hrapp IDENTIFIED by HRAPP;

User created.

SQL>
SQL> GRANT create session TO hrapp;

Grant succeeded.

SQL>
```

   b. Verify that `HRAPP` can connect. (Be aware of the uppercase password).

```
SQL> connect hrapp
Enter password: ******
```

```
Connected.
SQL>
SQL> EXIT
$
```

2. As the SEC user, drop the PFAY user to avoid possible conflicts. Then, create an end user with the following properties:

   Username: PFAY
   Password: oracle_4U
   PFAY is granted the create session privilege.
   PFAY can connect through HRAPP without a password.

3. For PFAY to connect through HRAPP, HRAPP must be a proxy. Use the GRANT CONNECT THROUGH syntax to allow HRAPP to proxy PFAY.

```
$ sqlplus sec
Enter password: ******
Last Successful login time: Mon Jun 17 2013 06:05:36 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL>
SQL> DROP USER pfay CASCADE;
DROP USER pfay CASCADE
          *
ERROR at line 1:
ORA-01918: user 'PFAY' does not exist



SQL> CREATE USER pfay IDENTIFIED by oracle_4U;

User created.

SQL> GRANT create session TO pfay;

Grant succeeded.

SQL> ALTER USER pfay  GRANT CONNECT THROUGH hrapp;

User altered.

SQL> EXIT
$
```

4. The `proxy_user` program tests connections through the middle tier.
   a. This program has the following arguments:
      Connection (TNS) name is required.
      Username is required.
      Password is optional.
   b. The program performs the following steps:
      1. Connects as the HRAPP user
      2. Creates a connection pool of 10 connections
      3. Creates 10 threads that connect to the database by using one of the connections from the pool. The `proxy_user` program makes these connections using the username and password parameters.
      4. Waits for a return character from the standard input
      5. Disconnects the 10 threads, destroys the connection pool, and ends
   c. Start a separate terminal window to act as a client. Set the environment variables by using the `oraenv` utility to set the instance name to `orcl`. Change to the `/home/oracle/labs/PROXY` directory.
   d. Recompile the proxy programs. Ignore the error messages.

```
$ cd /home/oracle/labs/PROXY
$ ./mk_proxy_user
proxy_user.c: In function 'main':
proxy_user.c:56: warning: incompatible implicit declaration of
built-in function 'strlen'
proxy_user.c: In function 'threadFunction':
proxy_user.c:109: warning: incompatible implicit declaration of
built-in function 'strlen'
$ ./mk_proxy_role
proxy_role.c: In function 'main':
proxy_role.c:60: warning: incompatible implicit declaration of
built-in function 'strlen'
proxy_role.c: In function 'threadFunction':
proxy_role.c:116: warning: incompatible implicit declaration of
built-in function 'strlen'
$ mv proxy_user? proxy_user
$ mv proxy_role? proxy_role
$
```

   e. Test the users that you created by executing `proxy_user` (from the operating system prompt) with the following command line:
      `$ ./proxy_user orcl pfay`
      where `orcl` is the TNS name for your local instance
      The `proxy_user` command connects `PFAY` without a password. Should this work? Why?
      The program should work because you set up `PFAY` so that the user can connect without a password. When the program is complete, press the Enter key.

```
$ ./proxy_user orcl pfay
Database: orcl
Username: pfay
Password:
Successful connection: Username: HRAPP
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Hit enter to end connections:
$
```

f. Examine the source code for the `proxy_user` program (see the appendix titled "Source Code").

5. Using the terminal window, select the information from the data dictionary that shows the users for whom `HRAPP` can proxy. Save this query; you will execute it again.

```
$ sqlplus sec
Enter password:  *******

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL>
SQL> COL proxy  FORMAT A6
SQL> COL client FORMAT A6
SQL> COL authentication FORMAT A12 WORD
SQL>
SQL> SELECT  proxy,
             client,
             authentication,
             authorization_constraint
      FROM dba_proxies
      WHERE proxy = 'HRAPP';
  2    3    4    5    6
PROXY  CLIENT AUTHENTICATI AUTHORIZATION_CONSTRAINT
------ ------ ------------ ----------------------------------
```

```
HRAPP  PFAY   NO           PROXY MAY ACTIVATE ALL CLIENT ROLES


SQL>
```

6. Modify the PFAY user so that a password is required when connecting through a middle tier.

```
SQL> ALTER USER pfay
       GRANT CONNECT THROUGH hrapp AUTHENTICATION REQUIRED;
  2
User altered.

SQL> exit
$
```

7. In the terminal window, run proxy_user with the following command line:

```
$ ./proxy_user orcl pfay
```

This command connects PFAY without a password. Should this work? Why?

Answer: The program should not work because the PFAY user now requires a password to connect.

```
$ ./proxy_user orcl pfay
Database: orcl
Username: pfay
Password:
Successful connection: Username: HRAPP
Error - ORA-28183: proper authentication not provided by proxy

Error - OCI_INVALID_HANDLE
Error - ORA-28183: proper authentication not provided by proxy

Error - OCI_INVALID_HANDLE
Error - ORA-28183: proper authentication not provided by proxy

Error - OCI_INVALID_HANDLE
Error - ORA-28183: proper authentication not provided by proxy

Error - OCI_INVALID_HANDLE
Error - ORA-28183: proper authentication not provided by proxy

Error - OCI_INVALID_HANDLE
Error - ORA-28183: proper authentication not provided by proxy

Error - OCI_INVALID_HANDLE
```

```
Error - ORA-28183: proper authentication not provided by proxy

Error - OCI_INVALID_HANDLE
Error - ORA-28183: proper authentication not provided by proxy

Error - OCI_INVALID_HANDLE
Error - ORA-28183: proper authentication not provided by proxy

Error - OCI_INVALID_HANDLE
Hit enter to end connections:
$
```

8. Run `proxy_user` with the following command line:

   ```
   $ ./proxy_user orcl pfay oracle_4U
   ```

   This command connects `PFAY` with a password. Should this work? Why?

   Answer: The program should work because the `PFAY` user now connects with a password.

   ```
   $ ./proxy_user orcl pfay oracle_4U
   Database: orcl
   Username: pfay
   Password: oracle_4U
   Successful connection: Username: HRAPP
   Successful connection: Username: pfay
   Successful connection: Username: pfay
   Successful connection: Username: pfay
   Successful connection: Username: pfay
   Successful connection: Username: pfay
   Successful connection: Username: pfay
   Successful connection: Username: pfay
   Successful connection: Username: pfay
   Successful connection: Username: pfay
   Successful connection: Username: pfay
   Hit enter to end connections:
   $
   ```

9. Select the information from the data dictionary that shows the users for whom `HRAPP` can proxy. (This is the same query as in step 5.) What is different from the query output in step 5?

   Answer: The `AUTHENTICATION` column values have changed to indicate that `PFAY` requires a password to connect.

   ```
   $ sqlplus sec
   Enter password: ******

   Connected to:
   ```

```
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL>
SQL> COL proxy  FORMAT A6
SQL> COL client FORMAT A6
SQL> COL authentication FORMAT A12 WORD
SQL>
SQL> SELECT  proxy,
             client,
             authentication,
             authorization_constraint
     FROM dba_proxies
     WHERE proxy = 'HRAPP';
  2    3    4    5    6
PROXY  CLIENT AUTHENTICATI AUTHORIZATION_CONSTRAINT
------ ------ ------------ ------------------------------
HRAPP  PFAY   YES          PROXY MAY ACTIVATE ALL CLIENT ROLES

SQL>
```

10. Change the `PFAY` user so that she can no longer connect through the middle tier.

```
SQL> ALTER USER pfay  REVOKE CONNECT THROUGH hrapp;

User altered.

SQL> exit
$
```

11. Run `proxy_user` with the following command:

    `$ ./proxy_user orcl pfay oracle_4U`

    This command connects `PFAY` with a password. Should this work? Why?

    Answer: The program works because the `PFAY` user connects with a password.

```
$ ./proxy_user orcl pfay oracle_4U
Database: orcl
Username: pfay
Password: oracle_4U
Successful connection: Username: HRAPP
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
```

```
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Hit enter to end connections:
$
```

12. Run `proxy_user` with the following command line:

    ```
    $ ./proxy_user orcl pfay
    ```

    This command connects `PFAY` without a password. Should this work? Why?

    The program should not work because the `PFAY` user requires a password to connect. Note that the error message is different from the message in step 7. Users do not require the `CONNECT THROUGH` privilege if they connect with a username and password.

```
$ ./proxy_user orcl pfay
Database: orcl
Username: pfay
Password:
Successful connection: Username: HRAPP
Error - ORA-01017: invalid username/password; logon denied

Error - OCI_INVALID_HANDLE
Error - ORA-01017: invalid username/password; logon denied

Error - OCI_INVALID_HANDLE
Error - ORA-01017: invalid username/password; logon denied

Error - OCI_INVALID_HANDLE
Error - ORA-01017: invalid username/password; logon denied

Error - OCI_INVALID_HANDLE
Error - ORA-01017: invalid username/password; logon denied

Error - OCI_INVALID_HANDLE
Error - ORA-01017: invalid username/password; logon denied

Error - OCI_INVALID_HANDLE
Error - ORA-01017: invalid username/password; logon denied
```

```
Error - OCI_INVALID_HANDLE
Error - ORA-01017: invalid username/password; logon denied

Error - OCI_INVALID_HANDLE
Error - ORA-01017: invalid username/password; logon denied

Error - OCI_INVALID_HANDLE
Hit enter to end connections:
$
```

13. Display the audited connections as the proxy user.

```
$ sqlplus / AS SYSDBA

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> COL dbusername FORMAT A10
SQL> COL dbproxy_username FORMAT A10
SQL> COL return_code FORMAT 999999
SQL> SELECT DISTINCT dbusername, dbproxy_username, return_code,
                     authentication_type
     FROM  unified_audit_trail
     WHERE dbproxy_username='HRAPP';
  2    3    4
DBUSERNAME DBPROXY_US RETURN_CODE
---------- ---------- -----------
AUTHENTICATION_TYPE
-----------------------------------------------------------------
-
PFAY    HRAPP        1017
(TYPE=(DATABASE));(CLIENT
ADDRESS=((ADDRESS=(PROTOCOL=tcp)(HOST=127.0.0.1)(PORT=
43150))));

PFAY    HRAPP       28183
(TYPE=(DATABASE));(CLIENT
ADDRESS=((ADDRESS=(PROTOCOL=tcp)(HOST=127.0.0.1)(PORT=
24516))));

PFAY    HRAPP       28183
(TYPE=(DATABASE));(CLIENT
ADDRESS=((ADDRESS=(PROTOCOL=tcp)(HOST=127.0.0.1)(PORT=
```

```
24513))));

PFAY   HRAPP       28183
(TYPE=(DATABASE));(CLIENT
ADDRESS=((ADDRESS=(PROTOCOL=tcp)(HOST=127.0.0.1)(PORT=
24443))));

PFAY   HRAPP         0
(TYPE=(PROXY));(CLIENT
ADDRESS=((ADDRESS=(PROTOCOL=tcp)(HOST=127.0.0.1)(PORT=242
83))));

PFAY   HRAPP        1017
(TYPE=(DATABASE));(CLIENT
ADDRESS=((ADDRESS=(PROTOCOL=tcp)(HOST=127.0.0.1)(PORT=
43157))));

… rows deleted

SQL> EXIT
$
```

Practices for Lesson 9: Using Privileges and Roles

Chapter 9

Practices for Lesson 9: Overview

Practices Overview

In these practices, the security officer will implement privileges and roles and grant them to users according to their respective job in the company.

Practice 9-1: Exploring DBA Privileges

Overview

In this practice, the security officer will manage the `DBA` role privileges in the non-CDB and in the PDBs of the CDB.

Tasks

1. Investigate the number of privileges of the DBA in the non-CDB.
   a. Use the `oraenv` utility to set the `ORACLE_SID` environment variable to the `orcl` value.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

   b. Connect as `SYSTEM` in `orcl` instance.

```
$ sqlplus system

Enter password: ******

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> SELECT * FROM session_roles ORDER BY 1;

ROLE
-----------------------------------------------------------------
-
AQ_ADMINISTRATOR_ROLE
CAPTURE_ADMIN
DATAPUMP_EXP_FULL_DATABASE
DATAPUMP_IMP_FULL_DATABASE
DBA
DELETE_CATALOG_ROLE
EM_EXPRESS_ALL
EM_EXPRESS_BASIC
EXECUTE_CATALOG_ROLE
EXP_FULL_DATABASE
GATHER_SYSTEM_STATISTICS
HS_ADMIN_EXECUTE_ROLE
```

```
HS_ADMIN_SELECT_ROLE
IMP_FULL_DATABASE
JAVA_ADMIN
JAVA_DEPLOY
OLAP_DBA
OLAP_XS_ADMIN
OPTIMIZER_PROCESSING_RATE
SCHEDULER_ADMIN
SELECT_CATALOG_ROLE
WM_ADMIN_ROLE
XDBADMIN
XDB_SET_INVOKER
XS_RESOURCE

25 rows selected.

SQL> SELECT * FROM session_privs ORDER BY 1;

PRIVILEGE
----------------------------------------
ADMINISTER ANY SQL TUNING SET
ADMINISTER DATABASE TRIGGER
ADMINISTER RESOURCE MANAGER
ADMINISTER SQL MANAGEMENT OBJECT
ADMINISTER SQL TUNING SET
ADVISOR
… rows deleted
UNLIMITED TABLESPACE
UPDATE ANY CUBE
UPDATE ANY CUBE BUILD PROCESS
UPDATE ANY CUBE DIMENSION
UPDATE ANY TABLE
USE ANY SQL TRANSLATION PROFILE

214 rows selected.

SQL>
```

Notice that the `SYSTEM` user is not granted the `SYSDBA` privilege.

c. Connect as `SYS` in `orcl` instance.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT * FROM session_roles ORDER BY 1;
```

```
no rows selected

SQL> SELECT * FROM session_privs ORDER BY 1;

PRIVILEGE
----------------------------------------
ADMINISTER ANY SQL TUNING SET
ADMINISTER DATABASE TRIGGER
… rows deleted
SYSDBA
SYSOPER
TRANSLATE ANY SQL
UNDER ANY TABLE
UNDER ANY TYPE
UNDER ANY VIEW
UNLIMITED TABLESPACE
UPDATE ANY CUBE
UPDATE ANY CUBE BUILD PROCESS
UPDATE ANY CUBE DIMENSION
UPDATE ANY TABLE
USE ANY SQL TRANSLATION PROFILE

233 rows selected.

SQL> EXIT
$
```

2. Now investigate if there are distinct DBAs for the `root` container and in the `pdb1_1` and `pdb1_2` containers in `cdb1` instance.
   a. Use the `oraenv` utility to set the `ORACLE_SID` environment variable to the `cdb1` value.

```
$ . oraenv
ORACLE_SID = [orcl] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

   b. Connect as `SYSTEM` in `cdb1` instance.

```
$ sqlplus system

Enter password: ******
Last Successful login time: Mon Jun 17 2013 05:38:37 +00:00
```

```
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL> col role format a30
SQL> SELECT role, common, con_id FROM cdb_roles
     WHERE role like '%DBA%';

ROLE                           COM  CON_ID
------------------------------ --- ------
DBA                            YES      3
CDB_DBA                        YES      3
PDB_DBA                        YES      3
XDBADMIN                       YES      3
OLAP_DBA                       YES      3
LBAC_DBA                       YES      3
DBA                            YES      2
CDB_DBA                        YES      2
PDB_DBA                        YES      2
XDBADMIN                       YES      2
OLAP_DBA                       YES      2
LBAC_DBA                       YES      2
DBA                            YES      1
CDB_DBA                        YES      1
PDB_DBA                        YES      1
XDBADMIN                       YES      1
OLAP_DBA                       YES      1
LBAC_DBA                       YES      1
DBA                            YES      4
CDB_DBA                        YES      4
PDB_DBA                        YES      4
XDBADMIN                       YES      4
OLAP_DBA                       YES      4
LBAC_DBA                       YES      4

24 rows selected.

SQL>
```

There are two types of DBA roles. The common DBA role systematically granted to any SYSTEM user created in a new PDB: the DBA role owns many system privileges. The common PDB_DBA role is also systematically granted to any SYSTEM user created in a new

PDB. The common PDB_DBA owns only three system privileges. In each PDB, the user being granted the DBA role, like the SYSTEM user, is able to grant distinct responsibilities to the administrators of the PDB he is responsible for.

```
SQL> COL username FORMAT A14
SQL> SELECT username, con_id
     FROM   cdb_users
     WHERE  username = 'SYSTEM' ;

USERNAME           CON_ID
-------------- ----------
SYSTEM                  1
SYSTEM                  4
SYSTEM                  3
SYSTEM                  2

SQL>
```

There are as many DBAs as containers: one for the root container and one DBA for each PDB.

c. Connect as the pdb1_1 DBA to create a junior DBA who you grant the local PDB_DBA role.

```
SQL> CONNECT system@pdb1_1
Enter password: ******
Connected.
SQL> COL grantee FORMAT A16
SQL> COL privilege FORMAT A26
SQL> SELECT * FROM dba_sys_privs WHERE grantee='PDB_DBA';

GRANTEE    PRIVILEGE                       ADM COM
---------------- -------------------------- --- ---
PDB_DBA    CREATE SESSION                  NO    NO
PDB_DBA    SET CONTAINER                   NO    NO
PDB_DBA    CREATE PLUGGABLE DATABASE       NO    NO

SQL> CREATE USER dba_junior IDENTIFIED BY oracle_4U;

User created.

SQL> GRANT create any table,
           create user, create role,
           create tablespace TO pdb_dba;
  2    3
Grant succeeded.
```

```
SQL> GRANT pdb_dba TO dba_junior;

Grant succeeded.

SQL> CONNECT dba_junior@pdb1_1
Enter password: ******
Connected.
SQL> SELECT * FROM session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
CREATE TABLESPACE
CREATE USER
CREATE ANY TABLE
CREATE ROLE
CREATE PLUGGABLE DATABASE
SET CONTAINER

7 rows selected.

SQL>
```

d. Connect as the `pdb1_2` DBA to create a junior DBA who you grant the local `PDB_DBA` role with different privileges.

```
SQL> CONNECT system@pdb1_2
Enter password: ******
Connected.
SQL> CREATE USER dba_junior IDENTIFIED BY oracle_4U;

User created.

SQL> GRANT create user, create role,
           create tablespace TO pdb_dba;
  2
Grant succeeded.

SQL> GRANT pdb_dba TO dba_junior;

Grant succeeded.

SQL> CONNECT dba_junior@pdb1_2
Enter password: ******
```

```
Connected.
SQL> SELECT * FROM session_privs;

PRIVILEGE
----------------------------
SET CONTAINER
CREATE PLUGGABLE DATABASE
CREATE ROLE
CREATE USER
CREATE TABLESPACE
CREATE SESSION

6 rows selected.

SQL> EXIT
$
```

Practice 9-2: Granting SYSBACKUP Administrative Privilege

Overview

In this practice, you manage the password file with the new 12 format dedicated to new administrative privileges like `SYSBACKUP`.

Tasks

1. Make sure you are in the `~/labs/PRIV` directory and your environment points to the `orcl` instance.

```
$ cd ~/labs/PRIV
$ . oraenv
ORACLE_SID = [cdb1] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

2. Run the `SYSBACKUP_setup.sh` script to recreate the password file.

```
$ ./SYSBACKUP_setup.sh
$
```

3. Connect with OS authentication with `AS SYSBACKUP` and check the user connected.

```
$ sqlplus / as sysbackup

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> show user
USER is "SYSBACKUP"
SQL>
```

4. List the privileges granted to `SYSBACKUP` user. Only a few privileges are granted to `SYSBACKUP` user. The `SYSBACKUP` privilege is granted to `SYSBACKUP` user.

```
SQL> select * from session_privs;

PRIVILEGE
----------------------------------------
SYSBACKUP
SELECT ANY TRANSACTION
SELECT ANY DICTIONARY
RESUMABLE
CREATE ANY DIRECTORY
ALTER DATABASE
```

```
AUDIT ANY
CREATE ANY CLUSTER
CREATE ANY TABLE
UNLIMITED TABLESPACE
DROP TABLESPACE
ALTER TABLESPACE
ALTER SESSION
ALTER SYSTEM

14 rows selected.

SQL>
```

5. Connect `AS SYSDBA` and list the privileges granted to `SYS` user. There are much more privileges granted to `SYS` user.

```
SQL> connect / as sysdba
Connected.
SQL> select * from session_privs;

PRIVILEGE
----------------------------------------
EXEMPT DDL REDACTION POLICY
EXEMPT DML REDACTION POLICY
LOGMINING
rows deleted …
AUDIT SYSTEM
ALTER SYSTEM

233 rows selected.

SQL>
```

6. Display from the `V$PWFILE_USERS` view. `SYS` user is the only user defined in the password file with `SYSDBA` and `SYSOPER` privileges only. `SYSBACKUP` user is not registered in the password file.

```
SQL> select * from v$pwfile_users;

USERNAME  SYSDB SYSOP SYSAS SYSBA SYSDG SYSKM CON_ID
--------  ----- ----- ----- ----- ----- ----- ------
SYS       TRUE  TRUE  FALSE FALSE FALSE FALSE     0

SQL>
```

7. Create a new user `JOHN` that will be granted the `SYSBACKUP` privilege in order to perform backup, restore, and recover operations, hence act as the `SYSBACKUP` user.

```
SQL> CREATE USER john IDENTIFIED BY oracle_4U;

User created.

SQL> GRANT create session, sysbackup TO john;
GRANT create session, sysbackup TO john
*
ERROR at line 1:
ORA-28017: The password file is in the legacy format.


SQL> EXIT
$
```

8. Because the password file had been created in legacy format, not compatible with the `SYSBACKUP` entry, it does not accept any `SYSBACKUP` entry.
   a. Recreate the file in 12 format, compatible with the `SYSBACKUP` entry.

```
$ cd $ORACLE_HOME/dbs
$ rm orapworcl
$ orapwd file=orapworcl password=oracle_4U entries=10 format=12
$
```

   b. Finally register `JOHN` in the password file.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> grant create session, SYSBACKUP to john;

Grant succeeded.

SQL> select * from v$pwfile_users;

USERNAME       SYSDB SYSOP SYSAS SYSBA SYSDG SYSKM     CON_ID
-------------- ----- ----- ----- ----- ----- ----- ----------
SYS            TRUE  TRUE  FALSE FALSE FALSE FALSE         0
JOHN           FALSE FALSE FALSE TRUE  FALSE FALSE         0

SQL>
```

   c. Attempt a remote connection in SQL*Plus.

```
SQL> connect john@orcl as SYSBACKUP
Enter password: ******
Connected.
SQL> SHOW USER
USER is "SYSBACKUP"
SQL> EXIT
$
```

d. Test the remote connection in RMAN.

```
$ rman target john/oracle_4U@orcl

Recovery Manager: Release 12.1.0.1.0 - Production on Mon Nov 26
06:28:43 2012

Copyright (c) 1982, 2012, Oracle and/or its affiliates.  All
rights reserved.

RMAN-00571: ===================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
RMAN-00571: ===================================================
RMAN-00554: initialization of internal recovery manager package
failed
RMAN-04005: error from target database:
ORA-01031: insufficient privileges
$
$ rman target '"john@orcl AS SYSBACKUP"'

target database Password: ******
connected to target database: ORCL (DBID=1345659572)
```

```
RMAN> select user from dual;

using target database control file instead of recovery catalog
USER
------------------------------
SYSBACKUP

RMAN> exit

Recovery Manager complete.
$
```

Practice 9-3: Implementing a Secure Application Role

Overview

This practice depends on Practices 4-1 and 8-1 for users and roles. It assumes that the SEC user has been created and granted certain privileges, and that the PFAY and HRAPP users have also been created.

Tasks

1. As the SEC user, create the HR_EMP_CLERK and HR_EMP_MGR roles. If you need to create the SEC user, use the /home/oracle/labs/USERS/create_sec.sh shell script.

```
$ sqlplus sec
Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> CREATE ROLE hr_emp_clerk;

Role created.

SQL> CREATE ROLE hr_emp_mgr;

Role created.

SQL>
```

2. Grant PFAY the HR_EMP_CLERK and HR_EMP_MGR roles. The PFAY user was created in Practice 8-1.

```
SQL> GRANT hr_emp_clerk, hr_emp_mgr TO pfay;

Grant succeeded.

SQL>
```

3. Give PFAY the ability to enable the HR_EMP_CLERK role through the HRAPP middle tier.

```
SQL> ALTER USER pfay
       GRANT CONNECT THROUGH hrapp
      WITH ROLE hr_emp_clerk;
 2   3

User altered.
```

```
SQL> EXIT
$
```

4. The `proxy_role` program enables roles through the middle tier. You simulate a middle tier by using a service name in the connect string. This program has the following arguments:

   Connection (TNS) name: Required

   Name of the role to be enabled: Required

   Username: Required

   Password: Optional

   The program performs the following steps:

   1) Connects as the HRAPP user
   2) Creates a connection pool of 10 connections
   3) Creates 10 threads that connect to the database by using one of the connections from the pool. The proxy_role program makes these connections using the username and password parameters.
   4) Enables the role for the user

   Test the user that you created by executing `proxy_role` (from the operating system prompt) with the following command line:

   ```
   $ /home/oracle/labs/PROXY/proxy_role orcl hr_emp_clerk pfay
   ```

   This command connects `PFAY` without a password and enables the `HR_EMP_CLERK` role. Should this work? Why?

   Be sure to use the name of your database instead of `orcl`. This works because `PFAY` can enable the `HR_EMP_CLERK` role through `HRAPP`.

   Note: Because each connection has its own thread, the following output is not sequential and the order of the output lines may differ for each execution.

```
$ /home/oracle/labs/PROXY/proxy_role orcl hr_emp_clerk pfay
Database: orcl
Role:     hr_emp_clerk
Username: pfay
Password:
Successful connection: Username: HRAPP
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Role successfully enabled: hr_emp_clerk
Successful connection: Username: pfay
Role successfully enabled: hr_emp_clerk
Successful connection: Username: pfay
Successful connection: Username: pfay
```

```
Role successfully enabled: hr_emp_clerk
Role successfully enabled: hr_emp_clerk
Role successfully enabled: hr_emp_clerk
Role successfully enabled: hr_emp_clerk
Role successfully enabled: hr_emp_clerk
Role successfully enabled: hr_emp_clerk
Role successfully enabled: hr_emp_clerk
Role successfully enabled: hr_emp_clerk
Hit enter to end connections:
$
```

5. Examine the source code for the `proxy_role` program (see the appendix titled "Source Code"). Execute `proxy_role` to enable the `HR_EMP_MGR` role for `PFAY`, using the following command line:

   ```
   $ /home/oracle/labs/PROXY/proxy_role orcl hr_emp_mgr pfay
   ```

   This command connects `PFAY` without a password and enables the `HR_EMP_MGR` role. Should this work? Why?

   Answer: It does not work. The reason is that `PFAY` does not have permission to enable the `HR_EMP_MGR` role through `HRAPP`.

```
$ /home/oracle/labs/PROXY/proxy_role orcl hr_emp_mgr pfay
Database: orcl
Role:     hr_emp_mgr
Username: pfay
Password:
Successful connection: Username: HRAPP
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Successful connection: Username: pfay
Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist

Successful connection: Username: pfay
Successful connection: Username: pfay
Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist

Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist

Successful connection: Username: pfay
Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist
```

```
Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist

Successful connection: Username: pfay
Successful connection: Username: pfay
Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist

Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist

Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist

Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist

Error - ORA-01924: role 'HR_EMP_MGR' not granted or does not
exist
Hit enter to end connections:
$
```

6. Select the information from the data dictionary that shows the users for whom `HRAPP` can proxy. What has changed?

   The `AUTHORIZATION_CONSTRAINT` column indicates that the proxy can only set some roles for the end user.

```
$ sqlplus sec
Enter password: ******
Connected.
SQL>
SQL> COL proxy  FORMAT A6
SQL> COL client FORMAT A6
SQL> COL authentication FORMAT A12 WORD
SQL>
SQL> SELECT  proxy,
             client,
             authentication,
             authorization_constraint
      FROM dba_proxies
      WHERE proxy = 'HRAPP';

PROXY  CLIENT AUTHENTICATI AUTHORIZATION_CONSTRAINT
------ ------ ------------ ----------------------------------
```

```
HRAPP   PFAY    NO            PROXY MAY ACTIVATE ROLE


SQL>
```

7. Look at the `tab_app_roles.sql` script. It creates a table similar to the one presented in the lesson, which is used to limit the IP addresses from which users can enable roles. Execute the script. Note that the `SEC` user connects through the listener. The `SEC.APP_ROLES` table is populated with the IP address of the current client IP address. The `SYS_CONTEXT('USERENV','IP_ADDRESS')` function is not populated unless the user connects through the listener. You must enter the net service name of your database. Enter the name of your database in the form of `orcl`. Remember that the password for `SEC` is `oracle_4sec`.

```
SQL> @/home/oracle/labs/PRIV/tab_app_roles.sql
SQL> CONNECT sec@orcl
Enter password: ******
Connected.
SQL>
SQL> ALTER USER sec DEFAULT TABLESPACE example QUOTA UNLIMITED
ON example;

User altered.

SQL>
SQL> DROP TABLE app_roles;
DROP TABLE app_roles
           *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> CREATE TABLE app_roles (id  NUMBER  CONSTRAINT app_roles_pk
PRIMARY KEY,
  username  VARCHAR2(30)NOT NULL, role  VARCHAR2(30), ip_address
VARCHAR2(15),
  CONSTRAINT app_roles_uk UNIQUE (username, role, ip_address));

Table created.

SQL> INSERT INTO app_roles
  2    VALUES (1, 'PFAY', 'HR_EMP_MGR',
  3      sys_context('userenv','ip_address'));

1 row created.

SQL> COMMIT;
```

```
Commit complete.

SQL>
```

8. As the SEC user, drop the HR_EMP_MGR role.

```
SQL> DROP ROLE hr_emp_mgr;

Role dropped.

SQL>
```

9. Create a secure application role with the following properties:

   Name: HR_EMP_MGR

   Enabled in the SEC.APP_ROLES_PKG package

```
SQL> CREATE ROLE hr_emp_mgr IDENTIFIED USING sec.app_roles_pkg;

Role created.

SQL>
```

10. Review the application code. How does it verify that the role can be enabled? Execute the application code.

```
set echo on
DROP PACKAGE app_roles_pkg;

CREATE OR REPLACE PACKAGE app_roles_pkg
  AUTHID CURRENT_USER
IS
  PROCEDURE set_role (
    p_role_name VARCHAR2 );
END;
/

CREATE OR REPLACE PACKAGE BODY app_roles_pkg IS
  PROCEDURE set_role (
    p_role_name VARCHAR2 )
  AS
    v_id  app_roles.id%TYPE;
  BEGIN
    SELECT  id
      INTO  v_id
      FROM  sec.app_roles
      WHERE username = sys_context('userenv','current_user')
        AND role = p_role_name
        AND ip_address = sys_context('userenv','ip_address');
```

```
      dbms_session.set_role(p_role_name);
    END;
  END;
  /
```

The role can be enabled if the role name, username, and IP address of the client are in the `APP_ROLES` table. This restricts which users can enable which roles from a particular client address.

```
SQL> set echo on
SQL>
SQL> DROP PACKAGE app_roles_pkg;
DROP PACKAGE app_roles_pkg
*
ERROR at line 1:
ORA-04043: object APP_ROLES_PKG does not exist


SQL>
SQL> CREATE OR REPLACE PACKAGE app_roles_pkg
      AUTHID CURRENT_USER
    IS
      PROCEDURE set_role (
        p_role_name VARCHAR2 );
    END;
    /
  2    3    4    5    6    7

Package created.

SQL>
SQL> CREATE OR REPLACE PACKAGE BODY app_roles_pkg IS
       PROCEDURE set_role (
         p_role_name VARCHAR2 )
       AS
         v_id  app_roles.id%TYPE;
       BEGIN
         SELECT  id
           INTO  v_id
           FROM  sec.app_roles
           WHERE username =
sys_context('userenv','current_user')
           AND role = p_role_name
           AND ip_address = sys_context('userenv','ip_address');
       dbms_session.set_role(p_role_name);
```

```
      END;
    END;
    /
 2    3    4    5    6    7    8    9   10   11   12   13   14
15   16
Package body created.

SQL>
```

11. As the SEC user, allow anyone to execute the SEC.APP_ROLES_PKG package and select from the SEC.APP_ROLES table. The user needs read access to the table because the package runs by using the privileges of the current user. What security problems does this create, and how can they be resolved?

```
SQL> GRANT execute ON app_roles_pkg TO public;

Grant succeeded.

SQL> GRANT select ON app_roles TO public;

Grant succeeded.

SQL>
```

12. Allowing anyone to execute the SEC.APP_ROLES_PKG package does not create any security problems because the appropriate row must appear in the APP_ROLES table before a role can be enabled. Giving read access to SEC.APP_ROLES allows any user to see which users can enable which roles from a client. If this is determined to be a security risk, you can create a view that shows only those rows that are related to the current user. The view would include the following predicate:

    ```
    WHERE username = sys_context('userenv','current_user')
    ```

    Test by performing the following steps:

    a. Connect as PFAY through the listener (you must use a service name orcl). Be sure to use your instance name instead of orcl.
    b. Query SESSION_ROLES to see which roles are enabled.
    c. Use the SEC.APP_ROLES_PKG package to enable the role.
    d. Query SESSION_ROLES to see which roles are enabled.

    Note: The HR_EMP_CLERK role that is enabled after the initial connection is from a previous step.

```
SQL> CONNECT pfay@orcl
Enter password: ******
Connected.
SQL>
SQL> SELECT * FROM session_roles;

ROLE
```

```
------------------------------
HR_EMP_CLERK

SQL>
SQL> EXEC sec.app_roles_pkg.set_role('HR_EMP_MGR');

PL/SQL procedure successfully completed.

SQL>
SQL> SELECT * FROM session_roles;

ROLE
------------------------------
HR_EMP_MGR

SQL>
```

13. What do you expect will happen if, as the `PFAY` user, you try to enable the `HR_EMP_MGR` role by using the `SET ROLE` command? Try it.

    Answer: It should return an error because it is a secure application role.

```
SQL> SET ROLE hr_emp_mgr;
SET ROLE hr_emp_mgr
*
ERROR at line 1:
ORA-28201: Not enough privileges to enable application role
'HR_EMP_MGR'

SQL>
```

14. As the `SEC` user, select the secure application role information from the data dictionary.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL>
SQL> COL role FORMAT A12
SQL> COL schema FORMAT A12
SQL> COL package FORMAT A30
SQL>
SQL> SELECT *
     FROM dba_application_roles
     WHERE ROLE = 'HR_EMP_MGR';
  2   3   4

ROLE          SCHEMA        PACKAGE
```

```
------------ ------------ ------------------------------
HR_EMP_MGR   SEC          APP_ROLES_PKG


SQL>
```

Practice 9-4: Enabling Roles at Run Time Using CBAC

Overview

In this practice, you will learn how to enable database roles at run time, enabling the procedure unit to execute with the required privileges in the calling user's environment. This is called CBAC (Code Based Access Control)

Tasks

1. Before testing the CBAC feature, execute the `CBAC_priv.sql` script. This script creates the end users `U1` and the schema `APP`, and the `APP.T1` table.

```
SQL> CONNECT / as sysdba
Connected.
SQL> @/home/oracle/labs/PRIV/CBAC_priv.sql
SQL> drop user u1 cascade;
drop user u1 cascade
          *
ERROR at line 1:
ORA-01918: user 'U1' does not exist

SQL> drop user app cascade;
drop user app cascade
          *
ERROR at line 1:
ORA-01918: user 'APP' does not exist

SQL>
SQL> create user u1 identified by oracle_4U default tablespace
users;

User created.

SQL> grant create session, create procedure to u1;

Grant succeeded.

SQL> create user app identified by oracle_4U default tablespace
users;

User created.

SQL> grant create session, create table, create procedure,
unlimited tablespace to app;

Grant succeeded.
```

```
SQL> create table app.T1 (code number);

Table created.

SQL> insert into app.T1 values (1);

1 row created.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL>
```

2. The `APP` schema creates two procedures: an invoker's right procedure, `IVPROC` and a definer's right procedure, `DFPROC`.
   a. Create the two procedures using the following codes:

```
CREATE OR REPLACE PROCEDURE app.ivproc (CODE in varchar2)
AUTHID CURRENT_USER AS
v_code  number;
BEGIN
SELECT code INTO v_code FROM app.t1;
dbms_output.put_line('Code is: '||v_code);
END ivproc;
/
```

```
SQL> CONNECT app
Enter password: ******
Connected.

SQL> CREATE OR REPLACE PROCEDURE app.ivproc (CODE in varchar2)
AUTHID CURRENT_USER AS
v_code  number;
BEGIN
SELECT code INTO v_code FROM app.t1;
dbms_output.put_line('Code is from Invoker right procedure:
'||v_code);
END ivproc;
/
2    3    4    5    6    7    8
Procedure created.
```

```
SQL>
```

b. Create the second procedure.

```
CREATE OR REPLACE PROCEDURE app.dfproc (CODE in varchar2)
AS
v_code  number;
BEGIN
SELECT code INTO v_code FROM app.t1;
dbms_output.put_line('Code is from Definer right procedure:
'||v_code);
END dfproc;
/
```

```
SQL> CREATE OR REPLACE PROCEDURE app.dfproc (CODE in varchar2)
AS
v_code  number;
BEGIN
SELECT code INTO v_code FROM app.t1;
dbms_output.put_line('Code is from Definer right procedure:
'||v_code);
END dfproc;
/
  2    3    4    5    6    7    8
Procedure created.

SQL>
```

3. You create the `ROLE1` role. Grant `SELECT` on `APP.T1` to the role. Create `ROLE2`. Grant `SELECT` on `SH.SALES` to the role and grant the role directly to the end user `U1`.

```
SQL> CONNECT / as sysdba
Connected.

SQL> CREATE ROLE role1;

Role created.

SQL> GRANT select ON APP.T1 to role1;

Grant succeeded.

SQL> CREATE ROLE role2;

Role created.
```

```
SQL> GRANT select ON SH.SALES to role2;

Grant succeeded.

SQL> GRANT role2 TO u1;

Grant succeeded.

SQL>
```

4. Grant the ROLE1 role to invoker's right procedure, IVPROC and to the definer's right procedure, DFPROC.

```
SQL> CONNECT app
Enter password: ******
Connected.
SQL> GRANT role1 TO PROCEDURE app.ivproc;
GRANT role1 TO PROCEDURE app.ivproc
                          *
ERROR at line 1:
ORA-01924: role 'ROLE1' not granted or does not exist

SQL>
```

5. Because the CBAC roles can only be granted to a program unit when the role is directly granted to the procedures' owner, grant the ROLE1 role to the APP procedures' owner.

```
SQL> CONNECT / as sysdba
Connected.
SQL> GRANT role1 TO app;

Grant succeeded.

SQL>
```

6. Now grant the role to the procedural units.

```
SQL> CONNECT app
Enter password: ******
Connected.
SQL> GRANT role1 TO PROCEDURE app.ivproc, PROCEDURE app.dfproc ;

Grant succeeded.

SQL>
```

7. Grant the EXECUTE privilege on both procedures to the U1 end user.

```
SQL> GRANT execute ON app.ivproc TO u1;
```

```
Grant succeeded.

SQL> GRANT execute ON app.dfproc TO u1;

Grant succeeded.

SQL>
```

8. Connect as `U1` and test how the CBAC enables roles at run time.
   a. Test the `app.ivproc` procedure.

```
SQL> CONNECT u1
Enter password: ******
Connected.
SQL> SELECT * FROM session_roles;

ROLE
-----------------------------------------------------------------
-
ROLE2

SQL> SET SERVEROUTPUT ON
SQL> EXEC app.ivproc(1)
Code is from Invoker right procedure: 1

PL/SQL procedure successfully completed.

SQL> SELECT * FROM session_roles;

ROLE
-----------------------------------------------------------------
-
ROLE2

SQL>
```

   Notice that the active role at login time is `ROLE2` only.

   b. Test the `app.dfproc` procedure.

```
SQL> EXEC app.dfproc(1)
Code is from Definer right procedure: 1

PL/SQL procedure successfully completed.

SQL> SELECT * FROM session_roles;
```

```
ROLE
--------------------------------------------------------------
ROLE2

SQL>
```

Notice that the execution completes as in 8.a.

c. Drop `ROLE1` and retest.

```
SQL> CONNECT system
Enter password: ******
Connected.
SQL> DROP ROLE role1;

Role dropped.

SQL> CONNECT u1
Enter password: ******
Connected.
SQL> SELECT * FROM session_roles;

ROLE
--------------------------------------------------------------
-
ROLE2

SQL> SET SERVEROUTPUT ON
SQL> EXEC app.ivproc(1)
BEGIN app.ivproc(1); END;

*
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at "APP.IVPROC", line 5
ORA-06512: at line 1

SQL> EXEC app.dfproc(1)
Code is from Definer right procedure: 1

PL/SQL procedure successfully completed.

SQL>
```

Practice 9-5: Executing Invoker's Right Procedure Using INHERIT PRIVILEGES Privilege *(Optional)*

Overview

In this practice you will use the new `INHERIT PRIVILEGES` privilege when creating invoker's rights procedures.

Tasks

1. Connected as `SYSTEM`, execute the inherit_priv.sql script to create `U1`, `U2` and `KATE` users and the `U2.T1` table.

```
SQL> CONNECT system
Enter password: ******
Connected.
SQL> @/home/oracle/labs/PRIV/inherit_priv.sql
SQL> drop user u1 cascade;

User dropped.

SQL> drop user u2 cascade;
drop user u2 cascade
          *
ERROR at line 1:
ORA-01918: user 'U2' does not exist


SQL> drop user kate;
drop user kate
          *
ERROR at line 1:
ORA-01918: user 'KATE' does not exist


SQL> create user kate identified by oracle_4U;

User created.

SQL> grant create session to kate;

Grant succeeded.

SQL> revoke INHERIT PRIVILEGES ON USER KATE from public;

Revoke succeeded.
```

```
SQL> create user u1 identified by oracle_4U default tablespace
users;

User created.

SQL> grant create session, create procedure to u1;

Grant succeeded.

SQL> create user u2 identified by oracle_4U default tablespace
users;

User created.

SQL> grant create session, create table, unlimited tablespace to
u2;

Grant succeeded.

SQL> create table u2.T1 (code number);

Table created.

SQL> insert into u2.T1 values (1);

1 row created.

SQL> commit;

Commit complete.

SQL> grant select on u2.T1 to u1;

Grant succeeded.

SQL> grant select on u2.T1 to kate;

Grant succeeded.

SQL>
SQL>
```

2. The developer `U1` creates an invoker's rights procedure that selects rows from `U2.T1` table. The user `U1` is granted the `SELECT` privilege on `U2.T1` table.
   a. Connect as user `U1`.

```
SQL> connect u1
Enter password: ******
Connected.
SQL>
```

   b. Create the `U1.PROC2` procedure.

```
CREATE OR REPLACE PROCEDURE u1.proc2 (CODE in varchar2)
AUTHID CURRENT_USER AS
v_code  number;
BEGIN
SELECT code INTO v_code FROM u2.t1;
dbms_output.put_line('Code is: '||v_code);
END PROC2;
/
```

```
SQL> CREATE OR REPLACE PROCEDURE u1.proc2 (CODE in varchar2)
AUTHID CURRENT_USER AS
v_code  number;
BEGIN
SELECT code INTO v_code FROM u2.t1;
dbms_output.put_line('Code is: '||v_code);
END PROC2;
/
  2    3    4    5    6    7    8
Procedure created.

SQL>
```

   c. Execute the procedure to test that it works successfully.

```
SQL> set serveroutput on
SQL> exec U1.PROC2('Code')
Code is: 1

PL/SQL procedure successfully completed.

SQL>
```

   d. The developer `U1` grants the `EXECUTE` privilege to the `KATE` user.

```
SQL> grant execute on U1.PROC2 to KATE;

Grant succeeded.

SQL>
```

3. KATE wants to test the procedure.
   a. KATE has no privilege on U2.T1 table. KATE connects and executes the procedure.

```
SQL> CONNECT kate
Enter password: ******
Connected.
SQL> set serveroutput on
SQL> exec U1.PROC2('Code')
BEGIN U1.PROC2('Code'); END;

*
ERROR at line 1:
ORA-06598: insufficient INHERIT PRIVILEGES privilege
ORA-06512: at "U1.PROC2", line 1
ORA-06512: at line 1

SQL>
```

   b. KATE grants the INHERIT PRIVILEGES on user KATE to procedure owner U1 thus allowing U1 to inherit her privileges during the execution of the procedure

```
SQL> grant INHERIT PRIVILEGES ON USER kate TO U1;

Grant succeeded.

SQL>
```

   c. KATE re-executes the procedure.

```
SQL> exec U1.PROC2('Code')
Code is: 1

PL/SQL procedure successfully completed.

SQL>
```

4. Display the users being granted the INHERIT PRIVILEGES privilege. There is a new object type 'USER' and the table name is the user name controlling who can access his privileges when he runs an invoker's rights procedure.

```
SQL> connect / as sysdba
Connected.

SQL> COL privilege FORMAT A20
SQL> COL type FORMAT A6
SQL> COL table_name FORMAT A10
SQL> COL grantee FORMAT A8
SQL> select PRIVILEGE, TYPE, TABLE_NAME, GRANTEE
     from DBA_TAB_PRIVS  where grantee='U1';
```

```
PRIVILEGE            TYPE   TABLE_NAME GRANTEE
-------------------- ------ ---------- --------
SELECT               TABLE  T1         U1
INHERIT PRIVILEGES   USER   KATE       U1

SQL>
```

5. Be aware that newly created users are granted the `INHERIT PRIVILEGES` privilege because the `INHERIT PRIVILEGES` privilege is granted to `PUBLIC`. The user `KATE` was revoked the `INHERIT PRIVILEGES` privilege at the beginning of the practice.
   a. Create a new user.

```
SQL> CREATE USER newuser IDENTIFIED BY newuser;

User created.

SQL>
```

   b. Check the privileges granted to `NEWUSER`.

```
SQL> select PRIVILEGE, TYPE, TABLE_NAME, GRANTEE
     from DBA_TAB_PRIVS
     where grantor='NEWUSER';
  2    3    4

PRIVILEGE            TYPE   TABLE_NAME GRANTEE
-------------------- ------ ---------- --------
INHERIT PRIVILEGES   USER   NEWUSER    PUBLIC

SQL> EXIT
$
```

Practice 9-6: BEQUEATH Current_user Views Using INHERIT PRIVILEGES *(Optional)*

Overview

In this practice you understand the different types of `BEQUEATH` views: the `CURRENT_USER` and the `DEFINER` views.

Assumption

The `bequeath_setup.sql` script is successfully completed.

Tasks

1. Make sure you are at the ~/labs/PRIV directory and your environment points to the `orcl` instance. Connect under `SYSTEM` user.

```
$ cd ~/labs/PRIV
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

2. Execute the `bequeath_setup.sql` script. The script creates users and grants appropriate privileges to the developer `U1` and the end user `KATE`.

```
$ sqlplus SYSTEM

Enter password: ******
Last Successful login time: Mon Jun 17 2013 09:51:24 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL>
SQL> @bequeath_setup.sql
Connected.
REVOKE select any table from OE
*
ERROR at line 1:
ORA-01952: system privileges not granted to 'OE'

User dropped.

User dropped.
```

```
User dropped.

User created.

Grant succeeded.

Revoke succeeded.

User created.

Grant succeeded.

SQL>
```

3. The developer U1 creates a BEQUEATH CURRENT_USER view. The view displays the current user connected.
   a. The user U1 connects and creates the view V_WHOAMI.

```
SQL> CONNECT u1
Enter password: ******
Connected.
SQL> CREATE OR REPLACE VIEW u1.v_whoami
     BEQUEATH CURRENT_USER
     AS SELECT ORA_INVOKING_USER "WHOAMI" FROM DUAL;
  2    3
View created.

SQL>
```

   b. The developer checks that the view V_WHOAMI works successfully.

```
SQL> select * from U1.V_WHOAMI;

WHOAMI
--------------------------------------------------------
U1

SQL>
```

4. The same developer U1 creates an BEQUEATH DEFINER view. The view displays the current user connected.
   a. The user U1 connects and creates the view V_WHOAMI_DEF.

```
SQL> CREATE OR REPLACE VIEW u1.v_whoami_def
     BEQUEATH DEFINER
     AS SELECT ORA_INVOKING_USER "WHOAMI" FROM DUAL;
  2    3
View created.
```

```
SQL>
```

b. The developer checks that the view `V_WHOAMI_DEF` works successfully.

```
SQL> select * from U1.V_WHOAMI_DEF;

WHOAMI
--------------------------------------------------------
U1

SQL>
```

5. The developer `U1` grants the `SELECT` privilege to `KATE` on both views.

```
SQL> grant SELECT  on U1.V_WHOAMI to KATE;

Grant succeeded.

SQL> grant SELECT  on U1.V_WHOAMI_DEF to KATE;

Grant succeeded.

SQL>
```

6. KATE connects and selects data from the BEQUEATH DEFINER view.

```
SQL> CONNECT kate
Enter password: ******
Connected.
SQL> select * from U1.V_WHOAMI_DEF;

WHOAMI
--------------------------------------------------------
KATE

SQL>
```

7. `KATE` selects data from the `BEQUEATH CURRENT_USER` view.

```
SQL> SELECT * FROM U1.V_WHOAMI;
select * from U1.V_WHOAMI
                 *
ERROR at line 1:
ORA-06598: insufficient INHERIT PRIVILEGES privilege

SQL>
```

8. `KATE` grants the `INHERIT PRIVILEGES ON USER KATE` to the view owner `U1`, allowing `U1` to use her privileges during the view execution.

```
SQL> grant INHERIT PRIVILEGES ON USER kate TO U1;

Grant succeeded.

SQL>
```

9. KATE attempts the statement on the BEQUEATH CURRENT_USER view.

```
SQL> select * from U1.V_WHOAMI;

WHOAMI
--------------------------------------------------------
KATE

SQL> EXIT
$
```

Practice 9-7: Managing Local and Common Privileges and Roles in CDB/PDBs

Overview

In this practice, you will grant local and common privileges, create and grant local and common roles in `cdb1` and in PDBs.

Assumptions

The following users have been successfully created from previous practice 6-5.

- `C##U1` common user in `cdb1`
- `LOCAL_EMPLOYEE` local user in `pdb1_1` (password `pass_pdb1`)
- `LOCAL_EMPLOYEE` local user in `pdb1_2` (password `pass_pdb2`)

Tasks

1. List all pre-defined roles in CDB.

```
$ . oraenv
ORACLE_SID = [orcl] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus  / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL> col role format a30
SQL> select ROLE, COMMON, CON_ID from cdb_roles order by role;

ROLE                           COM     CON_ID
------------------------------ --- ----------
ADM_PARALLEL_EXECUTE_TASK       YES        3
ADM_PARALLEL_EXECUTE_TASK       YES        4
ADM_PARALLEL_EXECUTE_TASK       YES        2
ADM_PARALLEL_EXECUTE_TASK       YES        1
APEX_ADMINISTRATOR_ROLE         YES        3
APEX_ADMINISTRATOR_ROLE         YES        4
APEX_ADMINISTRATOR_ROLE         YES        1
APEX_ADMINISTRATOR_ROLE         YES        2
…
DBA                            YES         3
DBA                            YES         4
```

```
DBA                            YES        2
DBA                            YES        1
…
XS_RESOURCE                     YES        3
XS_RESOURCE                     YES        4
XS_RESOURCE                     YES        1
XS_RESOURCE                     YES        2
XS_SESSION_ADMIN                YES        3
XS_SESSION_ADMIN                YES        1
XS_SESSION_ADMIN                YES        2
XS_SESSION_ADMIN                YES        4

337 rows selected.

SQL>
```

The common role is replicated in each container. The container ID 1 is the `root`. The container ID 2 is the `seed`. The container ID 3 is the `pdb1_1`. The container ID 4 is the `pdb1_2`.

2. View all common roles of the `root`.

```
SQL> select ROLE, COMMON from cdb_roles
     WHERE CON_ID = 1
     order by role;
  2    3
ROLE                            COM
------------------------------  ---
ADM_PARALLEL_EXECUTE_TASK       YES
APEX_ADMINISTRATOR_ROLE         YES
APEX_GRANTS_FOR_NEW_USERS_ROLE  YES
AQ_ADMINISTRATOR_ROLE                YES
AQ_USER_ROLE                    YES
AUDIT_ADMIN                     YES
AUDIT_VIEWER                    YES
…
CDB_DBA                         YES
CONNECT                         YES
…
DBA                             YES
…
XS_RESOURCE                     YES
XS_SESSION_ADMIN                YES

84 rows selected.
```

```
SQL>
```

Notice that all roles of the `root` are common: there cannot be any local roles in the `root`.

3. List all local roles in PDBs. The `HR_MGR` local role was created in practice 6-3 task 1.

```
SQL> SELECT role, con_id FROM CDB_ROLES
     WHERE common = 'NO' ;
  2
ROLE                                                   CON_ID
-------------------------------------------------- ----------
HR_MGR                                                      3

SQL>
```

4. Create a common C##_ROLE in `root`.

```
SQL> create role c##_role container=ALL;

Role created.

SQL>
```

5. Attempt to create a `LOCAL_ROLE` local role in `root`.

```
SQL> create role local_role container=CURRENT;
create role local_role container=CURRENT
*
ERROR at line 1:
ORA-65049: creation of local user or role is not allowed in
CDB$ROOT

SQL>
```

You get an error message because no local role is authorized in the `root`.

6. Create a common role in `pdb1_2`.

```
SQL> CONNECT system@pdb1_2
Enter password: ******
Connected.
SQL> CREATE ROLE c##_role_PDB1_2 container=ALL;
create role c##_role_PDB1_2 container=ALL
*
ERROR at line 1:
ORA-65050: Common DDLs only allowed in CDB$ROOT

SQL>
```

You get an error message because no common role can be created from a PDB.

7. Create a local role in `pdb1_2`.

```
SQL> CREATE ROLE local_role_PDB1_2 container=CURRENT;

Role created.

SQL> select ROLE, COMMON from dba_roles order by role;

ROLE                           COM
------------------------------ ---
ADM_PARALLEL_EXECUTE_TASK      YES
APEX_ADMINISTRATOR_ROLE        YES
…
C##_ROLE                       YES
CDB_DBA                        YES
CONNECT                        YES
…
DBA                            YES
…
LBAC_DBA                       YES
LOCAL_ROLE_PDB1_2              NO
…
PDB_DBA                        YES
…
XS_RESOURCE                    YES
XS_SESSION_ADMIN               YES

86 rows selected.

SQL>
```

8. Grant common or local roles as common or local.
   a. Grant a common role to a common user from the `root`.

```
SQL> connect / as sysdba
Connected.
SQL> grant c##_role to c##u1;

Grant succeeded.

SQL> col grantee format A16
SQL> col GRANTED_ROLE format A18
SQL> select GRANTEE, GRANTED_ROLE, COMMON, CON_ID
     from cdb_role_privs where grantee='C##U1';
  2
```

```
GRANTEE          GRANTED_ROLE     COM CON_ID
---------------- ---------------- --- ------
C##U1            C##_ROLE         NO       1

SQL>
```

Note that the common role is granted locally to the common user. The granted role is only applicable in the `root`.

```
SQL> connect c##u1
Enter password: ******
Connected.
SQL> select * from session_roles;

ROLE
------------------------------
C##_ROLE

SQL> connect c##u1@PDB1_2
Enter password: ******
Connected.
SQL> select * from session_roles;

no rows selected

SQL>
```

b. Now grant the common role to a common user from the `root` as common, to be applicable in all containers.

```
SQL> connect / as sysdba
Connected.
SQL> grant c##_role to c##u1 container=all;

Grant succeeded.

SQL>
```

```
SQL> col grantee format A16
SQL> col GRANTED_ROLE format A18
SQL> select GRANTEE, GRANTED_ROLE, COMMON, CON_ID
     from cdb_role_privs where grantee='C##U1';
  2
GRANTEE          GRANTED_ROLE      COM     CON_ID
---------------- ---------------- --- ----------
C##U1            C##_ROLE         NO            1
```

```
C##U1            C##_ROLE         YES           1
C##U1            C##_ROLE         YES           4
C##U1            C##_ROLE         YES           3

SQL> connect c##u1
Enter password: ******
Connected.
SQL> select * from session_roles;

ROLE
------------------------------
C##_ROLE

SQL> connect c##u1@PDB1_2
Enter password: ******
Connected.
SQL> select * from session_roles;

ROLE
------------------------------
C##_ROLE

SQL>
```

9. Revoke the common role from the common user so that the role cannot be used in any container.

```
SQL> connect / as sysdba
Connected.
SQL> revoke c##_role from c##u1 container=all;

Revoke succeeded.

SQL> connect c##u1
Enter password: ******
Connected.
SQL> select * from session_roles;
ROLE
------------------------------
C##_ROLE

SQL> connect c##u1@PDB1_2
Enter password: ******
Connected.
SQL> select * from session_roles;
```

```
no rows selected

SQL>
```

10. Grant a common role to a local user from the `root`.

```
SQL> connect / as sysdba
Connected.
SQL> grant c##_role to local_employee;
grant c##_role to local_employee
                     *
ERROR at line 1:
ORA-01917: user or role 'LOCAL_EMPLOYEE' does not exist

SQL>
```

Note that the user is unknown in `root`. It is a local user in `pdb1_2`.

11. Grant a common role to a local user in `pdb1_2`.

```
SQL> connect system@PDB1_2
Enter password: ******
Connected.
SQL> grant c##_role to local_employee;

Grant succeeded.

SQL> select GRANTEE, GRANTED_ROLE, COMMON, CON_ID
     from cdb_role_privs where grantee='LOCAL_EMPLOYEE';
  2
GRANTEE          GRANTED_ROLE      COM     CON_ID
---------------- ---------------- --- ----------
LOCAL_EMPLOYEE   C##_ROLE          NO           4

SQL>
```

Note that the user is granted a common role locally (common column = NO) applicable only in the `pdb1_2`.

12. Test the connection as the local user. The password is `pass_pdb2`.

```
SQL> connect local_employee@PDB1_2
Enter password: ******
Connected.
SQL> select * from session_roles;

ROLE
------------------------------
```

```
C##_ROLE

SQL>
```

13. Grant a common role to a local user from `pdb1_2` applicable in all containers.

```
SQL> connect system@PDB1_2
Enter password: ******
Connected.
SQL> grant c##_role to local_employee container=all;
grant c##_role to local_user_pdb2 container=all
*
ERROR at line 1:
ORA-65030: one may not grant a Common Privilege to a Local User
or Role

SQL>
```

Notice that a common role cannot be granted globally from a PDB.

14. Grant a local role to a local user from `pdb1_2`.

```
SQL> grant local_role_pdb1_2 to local_employee;

Grant succeeded.

SQL> select GRANTEE, GRANTED_ROLE, COMMON, CON_ID
     from cdb_role_privs where grantee='LOCAL_EMPLOYEE';
 2
GRANTEE          GRANTED_ROLE        COM     CON_ID
---------------- ------------------ --- ----------
LOCAL_EMPLOYEE   C##_ROLE             NO         4
LOCAL_EMPLOYEE   LOCAL_ROLE_PDB1_2    NO         4

SQL>
```

15. Test the connection as the local user.

```
SQL> connect  local_employee@PDB1_2
Enter password: ******
Connected.
SQL> select * from session_roles;

ROLE
------------------------------
C##_ROLE
LOCAL_ROLE_PDB1_2

SQL> EXIT
```

```
$
```

Practices for Lesson 10: Privilege Analysis

Chapter 10

Practices for Lesson 10: Overview

Practices Overview

In the practices for this lesson, you configure privileges, roles and contexts captures to make analyses of unnecessarily granted privileges.

Practice 10-1: Capturing Privileges

Overview

In this practice, you capture privileges used by users during a short period, generate the capture results, compare between used and unused privileges to decide which privileges might need to be revoked.

Tasks

1. Make sure you are at the `~/labs/PRIV` directory and your environment points to the `orcl` instance.

```
$ cd ~/labs/PRIV
$ . oraenv
ORACLE_SID = [cdb1] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

2. Run the `priv_setup.sql` script to create `JIM` and `TOM` users, `HR_MGR` and `SALES_CLERK` roles.

```
$ sqlplus system

Enter password: ******
Last Successful login time: Mon Jun 17 2013 09:59:11 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> @priv_setup.sql
Connected.

User dropped.


User created.


User dropped.


User created.
```

```
Grant succeeded.


drop role HR_MGR
          *
ERROR at line 1:
ORA-01919: role 'HR_MGR' does not exist


drop role SALES_CLERK
          *
ERROR at line 1:
ORA-01919: role 'SALES_CLERK' does not exist


drop role HR_MGR_JUNIOR
          *
ERROR at line 1:
ORA-01919: role 'HR_MGR_JUNIOR' does not exist


Role created.


Grant succeeded.


Grant succeeded.


Role created.


Grant succeeded.


Grant succeeded.

revoke select any table from oe
*
ERROR at line 1:
```

```
ORA-01952: system privileges not granted to 'OE'



User dropped.

drop user u2 cascade
          *
ERROR at line 1:
ORA-01918: user 'U2' does not exist



User dropped.


User created.


Grant succeeded.


Revoke succeeded.


User created.


Grant succeeded.


User created.


Grant succeeded.


Table created.


1 row created.
```

```
Commit complete.


Grant succeeded.


Grant succeeded.

SQL>
```

3. Define a capture of privileges used by all users. Use the following procedure.

```
exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
name            =>    'All_privs', -
description      =>  'All privs used', -
    type            =>    dbms_privilege_capture.g_database)
```

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
          name       =>   'All_privs', -
          description=>   'All privs used', -
           type       =>  dbms_privilege_capture.g_database)
> > >
PL/SQL procedure successfully completed.

SQL>
```

4. Start capturing the privileges while users are performing their daily work using privileges.
   a. Start the capture.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ( -
          name => 'All_privs')

PL/SQL procedure successfully completed.

SQL>
```

   b. Run the `priv_used_by_users.sql` script. The script connects as `JIM` who deletes rows from `HR.EMPLOYEES` table and `TOM` who selects rows from `SH.SALES` table.

```
SQL> @priv_used_by_users.sql
Connected.

24 rows deleted.

Commit complete.

Connected.

PROD_ID CUST_ID TIME_ID   CHANNEL_ID PROMO_ID QUANTITY_SOLD
AMOUNT_SOLD
------- ------- --------- ---------- -------- ------------- ----
------------
    120    6452 29-SEP-00          2      999             1
        6.4
    120    6452 29-SEP-00          4      999             1
        6.4

SQL>
```

5. Stop the capture.

```
SQL> connect system
Enter password: ******
Connected.
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ( -
          name      => 'All_privs')

PL/SQL procedure successfully completed.

SQL>
```

6. Generate the capture results. It may take a few minutes.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT ( -
```

```
          name => 'All_privs')

PL/SQL procedure successfully completed.

SQL>
```

7. Display the object privileges used during the capture period.

```
SQL> COL username FORMAT A10
SQL> COL object_owner FORMAT A12
SQL> COL object_name FORMAT A30
SQL> COL obj_priv FORMAT A25
SQL> SELECT  username, object_owner, object_name, obj_priv
     FROM  dba_used_objprivs
     WHERE username IN ('JIM', 'TOM');
  2    3
USERNAME   OBJECT_OWNER OBJECT_NAME                    OBJ_PRIV
---------- ------------ ------------------------------ -------
JIM        SYS          DUAL                           SELECT
JIM        SYSTEM       PRODUCT_PRIVS                  SELECT
TOM        SYS          ORA$BASE                       USE
TOM        SYSTEM       PRODUCT_PRIVS                  SELECT
JIM        SYS          DBMS_APPLICATION_INFO          EXECUTE
JIM        SYS          ORA$BASE                       USE
TOM        SYS          DUAL                           SELECT
TOM        SH           SALES                          SELECT
JIM        HR           EMPLOYEES                      DELETE
JIM        HR           EMPLOYEES                      SELECT
TOM        SYS          DBMS_APPLICATION_INFO          EXECUTE
JIM        SYS          DUAL                           SELECT
TOM        SYS          DUAL                           SELECT

13 rows selected.

SQL>
```

8. Display the system privileges used.

```
SQL> COL  sys_privs form a20
SQL> SELECT username, sys_priv FROM dba_used_sysprivs
     WHERE username IN ('JIM', 'TOM');
  2
USERNAME   SYS_PRIV
---------- --------------------
TOM        CREATE SESSION
JIM        CREATE SESSION

SQL>
```

9. Display the path of the privileges used if the privileges were granted to roles, and roles to users.

```
SQL> COL object_name FORMAT A10
```

```
SQL> COL path FORMAT A32
SQL> COL obj_priv FORMAT A10
SQL> SELECT username, obj_priv, object_name, path
     FROM   dba_used_objprivs_path
     WHERE  username IN ('TOM','JIM')
     AND object_name IN ('SALES','EMPLOYEES');
  2    3    4
USERNAME   OBJ_PRIV   OBJECT     PATH
---------- ---------- ---------- ---------------------------
TOM        SELECT     SALES      GRANT_PATH('TOM',
'SALES_CLERK')

JIM        DELETE     EMPLOYEES  GRANT_PATH('JIM', 'HR_MGR')

JIM        SELECT     EMPLOYEES  GRANT_PATH('JIM', 'HR_MGR')

SQL>
```

10. JIM is granted select, update, delete, insert privileges on HR.EMPLOYEES table through HR_MGR role. He used the DELETE and SELECT privileges until now.
The unused privileges are visible in DBA_UNUSED_PRIVS view.

```
SQL> SELECT username, sys_priv, obj_priv, object_name, path
     FROM   dba_unused_privs
     WHERE  username='JIM';

USERNAME SYS_PRIV OBJ_PRIV OBJECT    PATH
-------- -------- -------- --------- ---------------------------
-
JIM               INSERT   EMPLOYEES GRANT_PATH('JIM', 'HR_MGR')
JIM               UPDATE   EMPLOYEES GRANT_PATH('JIM', 'HR_MGR')

SQL>
```

11. Compare used and unused privileges. Finally you decide to revoke the INSERT privilege from JIM, but not impact other users who benefit from the HR_MGR role.
    a. You will first create a new role without the INSERT privilege and finally revoke the HR_MGR role from JIM.

```
SQL> create role HR_MGR_JUNIOR;

Role created.

SQL> GRANT select, update, delete ON hr.employees
     TO hr_mgr_junior;
 2
```

```
Grant succeeded.

SQL>
```

b. Grant the new role to `JIM`.

```
SQL> grant HR_MGR_JUNIOR to JIM;


Grant succeeded.


SQL>
```

c. Finally revoke the powerful privileged role `HR_MGR` from `JIM`.

```
SQL> revoke HR_MGR from JIM;

Revoke succeeded.


SQL>
```

12. Display the definition of the capture. The `ENABLED` column ensures that the `All_privs` capture has been stopped.

```
SQL> COL name FORMAT A12
SQL> COL type FORMAT A12
SQL> COL enabled FORMAT A2
SQL> COL roles FORMAT A26
SQL> COL context FORMAT a20
SQL> SELECT name, type, enabled,roles, context
     FROM dba_priv_captures;
  2
NAME         TYPE         EN ROLES                      CONTEXT
------------ ------------ -- -------------------------- --------
-
All_privs    DATABASE          N
SQL>
```

13. Delete the capture so as to remove all previous captured information from the views.

a. Execute the procedure.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE ( -
          name => 'All_privs')


PL/SQL procedure successfully completed.


SQL>
```

b. Verify that there is no data left of the `All_privs` capture.

```
SQL> SELECT username, sys_priv, obj_priv, object_name, path
     FROM   dba_unused_privs
     WHERE  username='JIM';
```

```
  2   3
no rows selected

SQL>
```

Practice 10-2: Capture Privileges Used Through Roles

Overview

In this practice, you capture the privileges used by roles during a short period, generate the capture results, compare between used and unused privileges to decide which privileges might need to be revoked.

Tasks

1. Define a capture of privileges used by roles `HR_MGR_JUNIOR` and `SALES_CLERK`. Use the following procedure.

```
exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
name           => 'Role_privs', -
description    => 'Privs used by HR_MGR_JUNIOR, SALES_CLERK', -
type           => dbms_privilege_capture.g_role, -
roles     => role_name_list('HR_MGR_JUNIOR', 'SALES_CLERK'))
```

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
name   => 'Role_privs', -
description => 'Privs used by HR_MGR_JUNIOR, SALES_CLERK', -
type        => dbms_privilege_capture.g_role, -
roles => role_name_list('HR_MGR_JUNIOR', 'SALES_CLERK'))
> > > >
PL/SQL procedure successfully completed.

SQL>
```

2. Start capturing the privileges while users perform their daily work.
   a. Start the capture.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ( -
          name => 'Role_privs')

PL/SQL procedure successfully completed.

SQL>
```

   b. Run the `priv_used_by_users.sql` script. The script connects as `JIM` who deletes rows from `HR.EMPLOYEES` table and `TOM` who selects rows from `SH.SALES` table.

```
SQL> @priv_used_by_users.sql
Connected.

0 rows deleted.

Commit complete.

Connected.
```

```
PROD_ID CUST_ID TIME_ID   CHANNEL_ID PROMO_ID QUANTITY_SOLD
AMOUNT_SOLD
------- ------- --------- ---------- -------- ------------- ----
------------
    120    6452 29-SEP-00          2      999             1
        6.4
    120    6452 29-SEP-00          4      999             1
        6.4

SQL>
```

3. Stop the capture.

```
SQL> connect system
Enter password: ******
Connected.
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ( -
          name      => 'Role_privs')

PL/SQL procedure successfully completed.

SQL>
```

4. Generate the capture results.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT ( -
          name => 'Role_privs')

PL/SQL procedure successfully completed.

SQL>
```

5. Display the object privileges used by the roles `HR_MGR_JUNIOR` and `SALES_CLERK` during the capture period.

```
SQL> col username FORMAT a8
SQL> col used_role FORMAT a20
SQL> col own FORMAT a4
SQL> SELECT  username, object_owner "OWN", object_name,
             obj_priv, used_role
     FROM    dba_used_objprivs
     WHERE   used_role IN ('HR_MGR_JUNIOR', 'SALES_CLERK');
  2    3
USERNAME OWN  OBJECT_NAME   OBJ_PRIV   USED_ROLE
-------- ---- -------------- ---------- --------------------
JIM      HR   EMPLOYEES     SELECT     HR_MGR_JUNIOR
TOM      SH   SALES         SELECT     SALES_CLERK
```

```
JIM       HR    EMPLOYEES     DELETE     HR_MGR_JUNIOR

SQL>
```

6. Display the system privileges used by the roles HR_MGR_JUNIOR and SALES_CLERK.

```
SQL> SELECT username, sys_priv, used_role
     FROM dba_used_sysprivs
     WHERE used_role IN ('HR_MGR_JUNIOR', 'SALES_CLERK');
  2    3
no rows selected

SQL>
```

7. HR_MGR_JUNIOR is granted select, update, delete on HR.EMPLOYEES table. The role used by JIM during the capture period used the DELETE and SELECT privileges until now. The unused privileges are visible in DBA_UNUSED_PRIVS view.

```
SQL> SELECT sys_priv, obj_priv, object_name, path
     FROM   dba_unused_privs
     WHERE  rolename IN ('HR_MGR_JUNIOR', 'SALES_CLERK');
  2    3
SYS_PRIV   OBJ_PRIV   OBJECT     PATH
---------- ---------- ---------- ----------------------------
           UPDATE     EMPLOYEES  GRANT_PATH('HR_MGR_JUNIOR')

SQL>
```

View the list of unused privileges: this list helps you decide whether to revoke or not the UPDATE privileges granted through the HR_MGR_JUNIOR role.

8. Display the definition of the capture. The ENABLED column shows that the Role_privs capture has been stopped. The numbers displayed in the roles list can be different from those here.

```
SQL> SELECT name, type, enabled,roles, context
     FROM dba_priv_captures;

NAME         TYPE         EN ROLES
------------ ------------ -- --------------------------
CONTEXT
----------------------------------------------------------------
Role_privs   ROLE         N  ROLE_ID_LIST(119, 115)

SQL>
```

9. Delete the capture so as to remove all previous captured information from the views.
   a. Execute the procedure.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE ( -
          name=> 'Role_privs')

PL/SQL procedure successfully completed.
```

```
SQL>
```

b. Verify that there is no data left of the `Role_privs` capture.

```
SQL> SELECT sys_priv, obj_priv, object_name, path
     FROM   dba_unused_privs
     WHERE  rolename IN ('HR_MGR_JUNIOR', 'SALES_CLERK');
  2   3
no rows selected

SQL>
```

Practice 10-3: Capture Privileges Used In Contexts *(Optional)*

Overview

In this practice, you capture privileges used by the user `TOM` or by the specific role `SALES_CLERK` during a short period, generate the capture results, compare between used and unused privileges to decide which privileges might need to be revoked.

Tasks

1. Define a capture of privileges used by the user `TOM` or by the specific role `SALES_CLERK`. Use the following procedure.

```
exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
name      => 'Special_capt', -
description  => 'Special', -
type      =>  dbms_privilege_capture.g_role_and_context, -
roles     =>   role_name_list('SALES_CLERK'), -
condition =>
'SYS_CONTEXT(''USERENV'',''SESSION_USER'')=''TOM''')
```

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
name        => 'Special_capt', -
description => 'Special', -
type    =>  dbms_privilege_capture.g_role_and_context, -
roles   =>  role_name_list('SALES_CLERK'), -
condition =>
'SYS_CONTEXT(''USERENV'',''SESSION_USER'')=''TOM''')
> > > > >
PL/SQL procedure successfully completed.

SQL>
```

2. Start capturing privileges while users perform their daily work using the privileges.
   a. Start the capture.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ( -
          name => 'Special_capt')

PL/SQL procedure successfully completed.

SQL>
```

   b. Run the `priv_used_by_users.sql` script. The script connects as `JIM` who deletes rows from `HR.EMPLOYEES` table and `TOM` who selects rows from `SH.SALES` table.

```
SQL> @priv_used_by_users.sql
Connected.

0 rows deleted.
```

```
Commit complete.

Connected.

PROD_ID CUST_ID TIME_ID   CHANNEL_ID PROMO_ID QUANTITY_SOLD
AMOUNT_SOLD
------- ------- --------- ---------- -------- ------------- ----
------------
    120    6452 29-SEP-00          2      999             1
        6.4
    120    6452 29-SEP-00          4      999             1
        6.4

SQL>
```

3. Stop the capture.

```
SQL> connect / as sysdba
Connected.
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ( -
          name      => 'Special_capt')

PL/SQL procedure successfully completed.

SQL>
```

4. Generate the capture results. It may take a few minutes.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT ( -
          name => 'Special_capt')

PL/SQL procedure successfully completed.

SQL>
```

5. Display the object privileges used.

```
SQL> SELECT  username, object_owner, object_name, obj_priv,
             used_role
     FROM  dba_used_objprivs
     WHERE username ='TOM' OR used_role='SALES_CLERK';
  2   3
USERNAME OBJECT_OWNER OBJECT_NAME    OBJ_PRIV   USED_ROLE
-------- ------------ -------------- ---------- --------------
TOM        SH         SALES            SELECT     SALES_CLERK

SQL>
```

6. Display the system privileges used.

```
SQL> SELECT username, sys_priv FROM dba_used_sysprivs;

no rows selected

SQL>
```

7. TOM is granted the select privilege on the SH.SALES table through SALES_CLERK role. He used the privilege.
   The unused privileges are visible in DBA_UNUSED_PRIVS view.
   There are no unused privileges. So there is no privilege that has been unnecessarily granted.

```
SQL> SELECT username, sys_priv, obj_priv, object_name, path
     FROM   dba_unused_privs
     WHERE  username='TOM' OR rolename='SALES_CLERK';
  2   3
no rows selected

SQL>
```

8. Delete the capture so as to remove all previous captured information from the views.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE ( -
          name=> 'Special_capt')

PL/SQL procedure successfully completed.

SQL> EXIT
$
```

Practices for Lesson 11: Using Application Contexts

Chapter 11

Practice 11-1: Creating an Application Context

In this practice, you create an application context, set the context using a secure package, and test the context.

Task

1. Match the following terms with their descriptions:

1. Namespace	A. An application context that is accessible only by the current session
2. Attribute	B. An application context whose values can be shared among sessions
3. `USERENV`	C. The identifier of an application context
4. Local	D. The built-in application context that contains information about the current session
5. Global	E. An application context that uses values from OID
6. Externalized context	F. Similar to a field. Its value can be modified only by the appropriate package.
7. Accessed globally	G. An application context that gets values from a source outside of the instance
8. `SYS_SESSION_ROLES`	H. The built-in application context that contains information about the enabled roles in the current session

1-C, 2-F, 3-D, 4-A, 5-E, 6-G, 7-B, 8-H

2. Connect as `PFAY` with the `oracle_4U` password and the `orcl` netservice. Using the `SYS_CONTEXT` procedure, display the following session-related attributes:

```
CURRENT_USER
SESSION_USER
PROXY_USER
IP_ADDRESS
NETWORK_PROTOCOL
AUTHENTICATION_TYPE
AUTHENTICATION_DATA
CLIENT_IDENTIFIER
EXTERNAL_NAME
```

3. You can use either of the following techniques to call `SYS_CONTEXT`:

```
SELECT sys_context('userenv','…')FROM dual;
EXEC dbms_output.put_line(syscontext('userenv','…'))
```

```
$ sqlplus pfay@orcl
```

```
Enter password: ******
Last Successful login time: Tue Jun 18 2013 00:22:32 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> SET SERVEROUTPUT ON
SQL>
SQL> SELECT sys_context('USERENV', 'CURRENT_USER') FROM DUAL;

SYS_CONTEXT('USERENV','CURRENT_USER')
-------------------------------------------------------------
PFAY

SQL> SELECT sys_context('USERENV', 'SESSION_USER') FROM DUAL;

SYS_CONTEXT('USERENV','SESSION_USER')
-------------------------------------------------------------
PFAY

SQL> SELECT sys_context('USERENV', 'PROXY_USER') FROM DUAL;

SYS_CONTEXT('USERENV','PROXY_USER')
-------------------------------------------------------------



SQL> SELECT sys_context('USERENV', 'IP_ADDRESS') FROM DUAL;

SYS_CONTEXT('USERENV','IP_ADDRESS')
-------------------------------------------------------------
127.0.0.1(loopback)

SQL> SELECT sys_context('USERENV', 'NETWORK_PROTOCOL') FROM
DUAL;

SYS_CONTEXT('USERENV','NETWORK_PROTOCOL')
-------------------------------------------------------------
tcp
```

```
SQL> SELECT sys_context('USERENV', 'AUTHENTICATION_TYPE') FROM
DUAL;

SYS_CONTEXT('USERENV','AUTHENTICATION_TYPE')
----------------------------------------------------------------
DATABASE

SQL> SELECT sys_context('USERENV', 'AUTHENTICATION_DATA') FROM
DUAL;

SYS_CONTEXT('USERENV','AUTHENTICATION_DATA')
----------------------------------------------------------------


SQL> SELECT sys_context('USERENV', 'CLIENT_IDENTIFIER') FROM
DUAL;

SYS_CONTEXT('USERENV','CLIENT_IDENTIFIER')
----------------------------------------------------------------


SQL> SELECT sys_context('USERENV', 'EXTERNAL_NAME') FROM DUAL;

SYS_CONTEXT('USERENV','EXTERNAL_NAME')
----------------------------------------------------------------

SQL>
```

If the user `PFAY` was a user known in an LDAP directory, the external name would display the DN known in the directory, like `'uid=pfay, ou=People, dc=example, dc=com'`. The session user would display `PFAY` being the global user name in the database.

```
SQL> EXEC dbms_output.put_line(sys_context( -
          'USERENV', 'CURRENT_USER'));
PFAY

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context( -
          'USERENV', 'SESSION_USER'));
PFAY

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context( -
```

```
          'USERENV', 'PROXY_USER'));

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context( -
          'USERENV', 'IP_ADDRESS'));
127.0.0.1(loopback)

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context( -
          'USERENV', 'NETWORK_PROTOCOL'));
tcp

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context( -
          'USERENV', 'AUTHENTICATION_TYPE'));
DATABASE

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context( -
          'USERENV', 'AUTHENTICATION_DATA'));

PL/SQL procedure successfully completed.

SQL>
```

If the user `PFAY` was a user known in an LDAP directory, the external name would display the DN known in the directory, like `'uid=pfay, ou=People, dc=example, dc=com'`.

4. The security officer grants new roles to `PFAY`. Use the built-in `SYS_SESSION_ROLES` context to indicate whether the roles are enabled after `PFAY`'s connection.
   Note: The `SEC` user was created in Practice 4-1, step 1.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> CREATE ROLE role_test;

Role created.

SQL> CREATE ROLE role_test2;
```

```
Role created.

SQL> GRANT role_test, role_test2 TO pfay;

Grant succeeded.

SQL> CONNECT pfay
Enter password: ******
Connected.
SQL> SELECT role FROM session_roles;

ROLE
--------
HR_EMP_CLERK
ROLE_TEST
ROLE_TEST2

SQL> SELECT SYS_CONTEXT('SYS_SESSION_ROLES', 'ROLE_TEST')
     FROM DUAL;
  2
SYS_CONTEXT('SYS_SESSION_ROLES','ROLE_TEST')
--------------------------------------------
TRUE

SQL> SELECT SYS_CONTEXT('SYS_SESSION_ROLES', 'DBA')
     FROM DUAL;
  2
SYS_CONTEXT('SYS_SESSION_ROLES','DBA')
--------------------------------------
FALSE

SQL> CONNECT sec
Enter password: ******
Connected.
SQL> DROP ROLE role_test;

Role dropped.

SQL> DROP ROLE role_test2;

Role dropped.
```

```
SQL>
```

5. Implement a local application context with the following properties:

   Name: `EMP_USER`

   Owned by: `SEC`

   This contains the following attributes, which are listed with the column from the `HR.EMPLOYEES` table that is used to obtain the attribute value:

Attribute	**Value**: **Column from `HR.EMPLOYEES`**
ID	EMPLOYEE_ID
NAME	FIRST_NAME || ' ' || LAST_NAME
EMAIL	EMAIL

```
SQL> CREATE CONTEXT emp_user USING current_emp;

Context created.

SQL>
```

6. The row in the `EMPLOYEES` table that is used to populate the attributes is selected by comparing the `EMAIL` column to the `SESSION_USER` attribute from `SYS_CONTEXT`.

   The procedure that sets the application context has the following properties:

   Owned by: `SEC` user

   Part of: `CURRENT_EMP` package

   Name: `SET_EMP_INFO`

   This is called from a logon trigger named `EMP_LOGON` that is also owned by `SEC`. This trigger applies to all users.

   You re-create a modified version of this package and context in a later practice, so save all your work.

   a. If you are not familiar with creating packages in PL/SQL, use the following code to create the package and package body:

```
CREATE OR REPLACE PACKAGE current_emp IS
     PROCEDURE set_emp_info;
     END;
/
CREATE OR REPLACE PACKAGE BODY current_emp IS
  PROCEDURE set_emp_info
  IS
    v_employee_id    hr.employees.employee_id%TYPE;
    v_first_name     hr.employees.first_name%TYPE;
    v_last_name      hr.employees.last_name%TYPE;
  BEGIN
    SELECT employee_id,
           first_name,
           last_name
      INTO v_employee_id,
```

```
            v_first_name,
            v_last_name
        FROM hr.employees
        WHERE email = SYS_CONTEXT('USERENV', 'SESSION_USER');
      DBMS_SESSION.SET_CONTEXT('emp_user', 'id',
        v_employee_id);
      DBMS_SESSION.SET_CONTEXT('emp_user', 'name',
        v_first_name || ' ' || v_last_name);
      DBMS_SESSION.SET_CONTEXT('emp_user', 'email',
        SYS_CONTEXT('USERENV', 'SESSION_USER'));
    EXCEPTION
      WHEN no_data_found THEN NULL;
    END;
  END;
  /
```

```
SQL> CREATE OR REPLACE PACKAGE current_emp IS
     PROCEDURE set_emp_info;
     END;
/
  2    3    4
Package created.

SQL>
```

```
SQL> CREATE OR REPLACE PACKAGE BODY current_emp IS
  PROCEDURE set_emp_info
  IS
    v_employee_id    hr.employees.employee_id%TYPE;
    v_first_name     hr.employees.first_name%TYPE;
    v_last_name      hr.employees.last_name%TYPE;
  BEGIN
    SELECT employee_id,
           first_name,
           last_name
      INTO v_employee_id,
           v_first_name,
           v_last_name
      FROM hr.employees
      WHERE email = SYS_CONTEXT('USERENV', 'SESSION_USER');
    DBMS_SESSION.SET_CONTEXT('emp_user', 'id',
      v_employee_id);
    DBMS_SESSION.SET_CONTEXT('emp_user', 'name',
      v_first_name || ' ' || v_last_name);
```

```
    DBMS_SESSION.SET_CONTEXT('emp_user', 'email',
      SYS_CONTEXT('USERENV', 'SESSION_USER'));
  EXCEPTION
    WHEN no_data_found THEN NULL;
  END;
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16   17   18   19   20   21   22   23   24   25   26
Package body created.

SQL>
```

b. Create the logon trigger.

```
SQL> CREATE or REPLACE TRIGGER emp_logon
  AFTER LOGON ON DATABASE
  BEGIN
  current_emp.set_emp_info;
  END;
/
  2    3    4    5    6
Trigger created.

SQL>
```

7. Test the context that you created by performing the following steps:
   a. Grant the `CREATE SESSION` privilege to the user named `SKING`.
   b. Log in as `SKING`.
   c. Use `SYS_CONTEXT` to verify that the `EMP_USER` context attributes are set. If you use `DBMS_OUTPUT`, remember to issue the `SET SERVEROUTPUT ON` command.

```
SQL> GRANT create session TO sking;

Grant succeeded.

SQL>
SQL> CONNECT sking
Enter Password: ******
Connected.
SQL>
SQL> SET SERVEROUTPUT ON
SQL> EXEC dbms_output.put_line(sys_context('emp_user', 'id'))
100

PL/SQL procedure successfully completed.
```

```
SQL> EXEC dbms_output.put_line(sys_context('emp_user', 'name'))
```

```
Steven King

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('emp_user', 'email'))
SKING

PL/SQL procedure successfully completed.

SQL>
```

8. Still connected as `SKING`, list all the application context attributes set in the current session. If Oracle Label Security is installed, the `LBAC$LABELS` and `LBAC$LASTSEQ` attributes are part of the context. It is populated because Oracle Label Security has been automatically configured when you executed the `/home/oracle/labs/DV/DV_setup.sh` script in practice 3-7 to configure and enable Database Vault. You disabled Database Vault by executing the `/home/oracle/labs/DV/DV_disable.sh` script but Oracle Label Security remains enabled.

```
SQL> DECLARE
  list dbms_session.AppCtxTabTyp;
  cnt number;
BEGIN
  dbms_session.list_context (list, cnt);
  IF cnt = 0
    THEN dbms_output.put_line('No contexts active.');
    ELSE
      FOR i IN 1..cnt LOOP
        dbms_output.put_line(list(i).namespace
          ||' ' || list(i).attribute
          || ' = ' || list(i).value);
      END LOOP;
  END IF;
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16
EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ = -1

PL/SQL procedure successfully completed.
```

```
SQL>
```

9. Log in as SEC and select information about the application context that you created from the data dictionary.

```
SQL> CONNECT sec
Enter Password: ******
Connected.
SQL> COL namespace FORMAT a10
SQL> COL schema FORMAT a8
SQL> COL package FORMAT a12
SQL> COL type FORMAT a20
SQL> SELECT * FROM dba_context WHERE namespace = 'EMP_USER';

NAMESPACE  SCHEMA   PACKAGE      TYPE
---------- -------- ------------ --------------------
EMP_USER   SEC      CURRENT_EMP  ACCESSED LOCALLY

SQL>
```

10. What happens when you call DBMS_SESSION.SET_CONTEXT to set an attribute in the EMP_USER context? Assume that SKING wants to change the context setting.

    Because the application context is set with a package, SKING does not have sufficient privileges to execute the DBMS_SESSION.SET_CONTEXT procedure.

```
SQL> CONNECT sking
Enter password:
Connected.
SQL> SET SERVEROUTPUT ON
SQL>
SQL> DECLARE
  list dbms_session.AppCtxTabTyp;
  cnt number;
BEGIN
  dbms_session.list_context (list, cnt);
  IF cnt = 0
    THEN dbms_output.put_line('No contexts active.');
    ELSE
      FOR i IN 1..cnt LOOP
        dbms_output.put_line(list(i).namespace
          ||' ' || list(i).attribute
          || ' = ' || list(i).value);
      END LOOP;
  END IF;
END;
```

```
/

EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ = -1

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_SESSION.SET_CONTEXT('emp_user', 'id', 1);
BEGIN DBMS_SESSION.SET_CONTEXT('emp_user', 'id', 1); END;

*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SYS.DBMS_SESSION", line 101
ORA-06512: at line 1


SQL> EXIT
$
```

Practices for Lesson 12: Implementing Virtual Private Database

Chapter 12

Practice 12-1: Implementing a Virtual Private Database Policy

Overview

In this practice, you create, enable, and test a fine-grained access control (FGAC) policy.

Task

1. How does FGAC determine which rows belong in the VPD for the current user?

   Fine-grained access control adds a predicate (condition) to the `WHERE` clause on a `SELECT` or DML statement with an `AND` operator.

2. How does FGAC know which tables are defined in the VPD?

   You include a table name or view name when the fine-grained access control policy is created.

3. In this practice, you implement a security policy that allows users to see only their own rows in the `HR.EMPLOYEES` table. The practice uses the `SEC` and `SKING` users, and the application context created in the lesson titled "Using Application Contexts." If you did not complete that practice, execute the following scripts after connecting to the database `AS SYSDBA`:

   `~/labs/USERS/create_sec.sql` creates the `SEC` user.

   `~/labs/VPD/create_context.sql` creates the `EMP_USER` application context.

   `~/labs/VPD/create_pack_trig.sql` creates the packages and trigger.

   `~/labs/VPD/create_SKING.sql` creates the `SKING` user and tests the application context.

   Your output may vary depending on which objects already exist in the database; however, you should not receive any errors on the `CREATE` statements.

4. The `SEC` user also needs the privilege to create policies. Grant `SEC` the ability to execute the package that creates policies.

```
$ sqlplus / AS SYSDBA


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL>
SQL> GRANT execute ON dbms_rls TO sec;


Grant succeeded.

SQL>
```

5. What privilege exempts the user from access policies? Why does the `SEC` user need this privilege? Grant it to `SEC`.

   The `EXEMPT ACCESS POLICY` privilege is very powerful. Statements that are issued by a user with this privilege do not have any FGAC policies applied. This privilege can also be granted by `SYSTEM`.

```
SQL> GRANT exempt access policy TO sec;

Grant succeeded.

SQL>
```

6. Create the package that is used by the security policy to return a predicate.
   a. Create the package specification.

```
SQL> CONNECT sec
Enter Password: ******
Connected.
SQL>
SQL> CREATE OR REPLACE PACKAGE hr_policy_pkg IS
FUNCTION limit_emp_emp (
             object_schema IN VARCHAR2,
             object_name      VARCHAR2 )
RETURN VARCHAR2;
END;
/
  2    3    4    5    6    7
Package created.

SQL>
```

   b. Create the package body.

```
SQL> CREATE OR REPLACE PACKAGE BODY hr_policy_pkg IS
FUNCTION limit_emp_emp (
             object_schema IN  VARCHAR2,
             object_name       VARCHAR2 )
RETURN VARCHAR2
IS
  v_emp_id NUMBER;
  BEGIN
   RETURN 'employee_id = SYS_CONTEXT(''emp_user'', ''id'')';
 END;
END;
/
  2    3    4    5    6    7    8    9   10   11   12
Package body created.

SQL>
```

   c. What predicate does the policy use to limit the rows returned from the `EMPLOYEE` table?

      `employee_id = SYS_CONTEXT('emp_user', 'id')`

   d. How does this predicate limit the rows?

The user making the query must have an `EMAIL_ID` that matches the database username, and the `emp_user` attribute in `sys_context` is set equal to the `employee_id` of the user (see Practice 11-1, step 5). The predicate allows the user to access only the record describing the user.

7. Test the policy function.

```
SQL> SELECT hr_policy_pkg.limit_emp_emp('a', 'b') FROM DUAL;

HR_POLICY_PKG.LIMIT_EMP_EMP('A','B')
------------------------------------------------------------
employee_id = SYS_CONTEXT('emp_user', 'id')

SQL>
```

8. Implement a policy with the following characteristics:

   The policy limits the rows that are selected from the `HR.EMPLOYEES` table.

   The policy is named `HR_EMP_POL`.

   The function that is used to return a predicate is `SEC.HR_POLICY_PKG.LIMIT_EMP_EMP`.

```
SQL> EXEC dbms_rls.drop_policy('HR', 'EMPLOYEES','HR_EMP_POL')
BEGIN dbms_rls.drop_policy('HR', 'EMPLOYEES', 'HR_EMP_POL')
END;

*
ERROR at line 1:
ORA-28102: policy does not exist
ORA-06512: at "SYS.DBMS_RLS", line 126
ORA-06512: at line 1

SQL> EXEC dbms_rls.add_policy('HR','EMPLOYEES', -
            'HR_EMP_POL','SEC', -
            'HR_POLICY_PKG.LIMIT_EMP_EMP','SELECT')

PL/SQL procedure successfully completed.

SQL>
```

9. Set up the `SKING` user so that he can access the `HR.EMPLOYEES` table. Because `SEC` has `GRANT ANY OBJECT PRIVILEGE`, the `SEC` user can grant this privilege. Grant the same privilege to `PFAY`.

```
SQL> GRANT select ON hr.employees TO sking;

Grant succeeded.

SQL> GRANT select ON hr.employees TO pfay;
```

```
Grant succeeded.

SQL>
```

10. As SKING, display the current context attributes.

```
SQL> connect sking
Enter Password: ******
Connected.
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  list dbms_session.AppCtxTabTyp;
  cnt number;
BEGIN
  dbms_session.list_context (list, cnt);
  IF cnt = 0
    THEN dbms_output.put_line('No contexts active.');
    ELSE
      FOR i IN 1..cnt LOOP
        dbms_output.put_line(list(i).namespace
          ||' ' || list(i).attribute
          || ' = ' || list(i).value);
      END LOOP;
  END IF;
END;
/

EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ = -1

PL/SQL procedure successfully completed.

SQL>
```

11. Which rows are returned when SKING queries the HR.EMPLOYEES table without a WHERE clause? Try it.

```
SQL> select employee_id, first_name, last_name, email
     from HR.EMPLOYEES;
  2
EMPLOYEE_ID FIRST_NAME      LAST_NAME         EMAIL
----------- --------------- ----------------- -----------------
        100 Steven          King              SKING
```

```
SQL>
```

12. Which rows are returned when `PFAY` queries the `HR.EMPLOYEES` table without a `WHERE` clause? Try it.

```
SQL> CONNECT pfay
Enter Password: ******
Connected.
SQL> select employee_id, first_name, last_name, email
     from HR.EMPLOYEES;
  2
EMPLOYEE_ID FIRST_NAME      LAST_NAME         EMAIL
----------- --------------- ----------------- -----------------
        202 Pat             Fay               PFAY

SQL>
```

13. Sometimes, it is necessary to view the predicate that is added by the policy.
    a. Connect as `SEC` to view the predicate added by the policy and use the views `V$VPD_POLICY` and `V$SQL`.

```
SQL> CONNECT sec
Enter Password: ******
Connected.
SQL> SELECT distinct policy, predicate, sql_text
     FROM   v$vpd_policy p,  v$sql s
     WHERE  s.child_address = p.address;
  2    3
POLICY    PREDICATE
--------- ------------------------------------------------------
SQL_TEXT
-------------------------------------------------------------------
----------------
HR_EMP_POL
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME FROM HR.EMPLOYEES
WHERE EMAIL = SYS_CONTEXT('USERENV', 'SESSION_USER')

HR_EMP_POL employee_id = SYS_CONTEXT('emp_user', 'id')
select employee_id, first_name, last_name, email      from
HR.EMPLOYEES

SQL>
```

    b. You can also use SQL tracing. The user must have the `ALTER SESSION` privilege to turn on this type of tracing. `SYS` has the ability to grant this privilege, but this ability has not been granted to `SEC`. To enable a trace that will capture the predicate, execute the following command:

```
ALTER SESSION SET EVENTS '10730 TRACE NAME CONTEXT FOREVER,
LEVEL 1';
```

Grant `SKING` the `ALTER SESSION` privilege, and then capture the predicate in a trace file.

```
SQL> connect / as sysdba
Connected.
SQL> GRANT ALTER SESSION TO SKING;

Grant succeeded.

SQL> connect SKING
Enter password: ******
Connected.
SQL> ALTER SESSION SET EVENTS '10730 TRACE NAME CONTEXT FOREVER,
LEVEL 1';

Session altered.

SQL> SELECT employee_id, first_name, last_name, email
     FROM hr.employees;

EMPLOYEE_ID FIRST_NAME     LAST_NAME       EMAIL
----------- -------------- --------------- -------------------
        100 Steven         King            SKING

SQL> EXIT
$
```

14. View the trace file. The trace file will be created in the Automatic Diagnostics Directory by default. Look for the file in the `$ORACLE_BASE/diag/rdbms/orcl/orcl/trace` directory.

    Hint: The `ls -ltr` command lists the trace files in reverse order by time, so the most recent files will be at the end of the listing. Also, the trace file will have a `.trc` extension.

```
$ cd $ORACLE_BASE/diag/rdbms/orcl/orcl/trace
$ ls -ltr *ora*.trc
…
lines deleted
…
-rw-r----- 1 oracle oinstall   915 Apr 25 03:03
orcl_ora_11899.trc
-rw-r----- 1 oracle oinstall  1033 Apr 25 05:43
orcl_ora_2762.trc
-rw-r----- 1 oracle oinstall  1348 Apr 25 06:06
orcl_ora_5814.trc
$
```

```
$ cat orcl_ora_5814.trc
Trace file
/u01/app/oracle/diag/rdbms/orcl/orcl/trace/orcl_ora_5814.trc
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1
System name:    Linux
Node name:      EDRSR32P1
Release:  2.6.39-200.24.1.el6uek.x86_64
Version:  #1 SMP Sat Jun 23 02:39:07 EDT 2012
Machine:  x86_64
Instance name: orcl
Redo thread mounted by this instance: 1
Oracle process number: 41
Unix process pid: 5814, image: oracle@EDRSR32P1 (TNS V1-V3)


*** 2013-04-25 06:06:31.769
*** SESSION ID:(275.3461) 2013-04-25 06:06:31.769
*** CLIENT ID:() 2013-04-25 06:06:31.769
*** SERVICE NAME:(SYS$USERS) 2013-04-25 06:06:31.769
*** MODULE NAME:(SQL*Plus) 2013-04-25 06:06:31.769
*** ACTION NAME:() 2013-04-25 06:06:31.769

---------------------------------------------------------------
Logon user     : SKING
Table/View     : HR.EMPLOYEES
VPD Policy name: HR_EMP_POL
Policy function: SEC.HR_POLICY_PKG.LIMIT_EMP_EMP
RLS view :
SELECT
"EMPLOYEE_ID","FIRST_NAME","LAST_NAME","EMAIL","PHONE_NUMBER","H
IRE_DATE","JOB_ID","SALARY","COMMISSION_PCT","MANAGER_ID","DEPAR
TMENT_ID" FROM "HR"."EMPLOYEES"   "EMPLOYEES" WHERE (employee_id
= SYS_CONTEXT('emp_user', 'id'))
---------------------------------------------------------------
$
```

15. Using Enterprise Manager Cloud Control, delete the `HR_EMP_POL` fine-grained access control policy.

Step	Page	Action
a.		In the browser, enter the following URL:

		https://localhost:7802/em
	Login	Enter: User Name: `sysman` Password: `Oracle123`
	Enterprise Summary	Click the **Targets** tab, then the **Databases** option.
b.	Databases	Click the **`orcl`** link.
c.	orcl	Click the **Administration** tab, then the **Security** option, then the **Virtual Private Database** option.
	Database Login	Click **Login**. Use `CREDORCL` credentials to login.
e.	Virtual Private Database Policies	Select the `HR_EMP_POL` policy. Click **Delete**.
f.	Confirmation	Click **Yes**.
g.	Virtual Private Database Policies	You receive the following message: Update Message: `POLICY HR_EMP_POL` has been deleted successfully

16. Change the security policy to allow everyone to view the `HR.EMPLOYEES` table, but not the `SALARY` and `COMMISSION_PCT` columns. The `HR.EMPLOYEES` table can then be used as a phone directory.
    Create the new policy defining the two following parameters: `SEC_RELEVANT_COLS` and `SEC_RELEVANT_COL_OPTS`.

```
$ sqlplus sec

Enter password: ******
Last Successful login time: Tue Jun 18 2013 00:49:49 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL>
SQL> BEGIN
  dbms_rls.add_policy(object_schema => 'HR',
  object_name => 'EMPLOYEES',
  policy_name => 'HR_EMP_POL',
  function_schema => 'SEC',
  policy_function => 'HR_POLICY_PKG.LIMIT_EMP_EMP',
  statement_types =>'SELECT',
  sec_relevant_cols => 'SALARY,COMMISSION_PCT',
  sec_relevant_cols_opt => dbms_rls.ALL_ROWS);
 END;
```

```
/
2     3     4     5     6     7     8     9    10    11
PL/SQL procedure successfully completed.

SQL>
```

17. Test this new policy with the SKING user. Note that in the first SELECT statement, all the rows and columns that are requested are shown. In the second SELECT statement, SKING sees his own salary but no other salary is displayed. Set tracing so that you can view the changed SQL statement later.

```
SQL> connect sking
Enter password: *****
Connected.
SQL> COL first_name FORMAT A12
SQL> COL LAST_NAME FORMAT A12
SQL> ALTER SESSION SET EVENTS '10730 TRACE NAME CONTEXT FOREVER,
LEVEL 1';

Session altered.

SQL> select first_name, last_name, email from hr.employees;

FIRST_NAME   LAST_NAME    EMAIL
------------ ------------ -------------------------
Ellen        Abel         EABEL
Sundar       Ande         SANDE
David        Austin       DAUSTIN
Hermann      Baer         HBAER
Amit         Banda        ABANDA
… rows deleted …
Clara        Vishney      CVISHNEY
Shanta       Vollman      SVOLLMAN
Alana        Walsh        AWALSH
Matthew      Weiss        MWEISS
Jennifer     Whalen       JWHALEN
Eleni        Zlotkey      EZLOTKEY

83 rows selected.

SQL> select first_name, last_name, SALARY, COMMISSION_PCT
     from hr.employees;

FIRST_NAME   LAST_NAME        SALARY COMMISSION_PCT
------------ ------------ ---------- --------------
```

```
Steven       King              24000
Neena        Kochhar
Lex          De Haan
Alexander    Hunold
… rows deleted …
Hermann      Baer
Shelley      Higgins
William      Gietz

83 rows selected.

SQL> EXIT
$
```

18. View the trace file and note the change to the SQL statements. A `CASE` clause is added to the `SELECT` clause for each relevant column.

```
$ cd $ORACLE_BASE/diag/rdbms/orcl/orcl/trace
$ ls -ltr *ora*.trc
…
lines deleted
…
rw-r----- 1 oracle oinstall  1272 Apr 25 07:44
orcl_ora_19917.trc
-rw-r----- 1 oracle oinstall  1262 Apr 25 07:45
orcl_ora_20091.trc
-rw-r----- 1 oracle oinstall   914 Apr 25 07:46
orcl_ora_20160.trc
-rw-r----- 1 oracle oinstall  2132 Apr 25 07:51
orcl_ora_20858.trc
$
$ cat orcl_ora_20858.trc
...

---------------------------------------------------------------
Logon user     : SKING
Table/View     : HR.EMPLOYEES
VPD Policy name    : HR_EMP_POL
Policy function: SEC.HR_POLICY_PKG.LIMIT_EMP_EMP
RLS view :
SELECT
"EMPLOYEE_ID","FIRST_NAME","LAST_NAME","EMAIL","PHONE_NUMBER","H
IRE_DATE","JOB_ID", CASE WHEN (employee_id =
SYS_CONTEXT('emp_user', 'id')) THEN "SALARY" ELSE NULL END
"SALARY", CASE WHEN (employee_id = SYS_CONTEXT('emp_user',
'id')) THEN "COMMISSION_PCT" ELSE NULL END
```

```
"COMMISSION_PCT","MANAGER_ID","DEPARTMENT_ID" FROM
"HR"."EMPLOYEES"   "EMPLOYEES"
-------------------------------------------------------------


*** 2013-04-25 08:02:13.317
-------------------------------------------------------------
Logon user     : SKING
Table/View     : HR.EMPLOYEES
VPD Policy name    : HR_EMP_POL
Policy function: SEC.HR_POLICY_PKG.LIMIT_EMP_EMP
RLS view :
SELECT
"EMPLOYEE_ID","FIRST_NAME","LAST_NAME","EMAIL","PHONE_NUMBER","H
IRE_DATE","JOB_ID", CASE WHEN (employee_id =
SYS_CONTEXT('emp_user', 'id')) THEN "SALARY" ELSE NULL END
"SALARY", CASE WHEN (employee_id = SYS_CONTEXT('emp_user',
'id')) THEN "COMMISSION_PCT" ELSE NULL END
"COMMISSION_PCT","MANAGER_ID","DEPARTMENT_ID" FROM
"HR"."EMPLOYEES"   "EMPLOYEES"
-------------------------------------------------------------
$
```

19. Clean up after this practice by dropping the policy.

```
$ sqlplus sec
Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options


SQL> EXEC dbms_rls.drop_policy('HR', 'EMPLOYEES','HR_EMP_POL')


PL/SQL procedure successfully completed.


SQL>
```

Practice 12-2: Implementing a Dynamic VPD Policy

Overview

In this practice, you will find out how setting the wrong type for your VPD policy leads to wrong results.

Tasks

1. Create a static policy. The policy calls a function displaying rows in a table depending on the time.

```
SQL> exec DBMS_RLS.DROP_POLICY ('HR', 'EMPLOYEES','POL_TIME');
BEGIN DBMS_RLS.DROP_POLICY ('HR', 'EMPLOYEES','POL_TIME'); END;

*
ERROR at line 1:
ORA-28102: policy does not exist
ORA-06512: at "SYS.DBMS_RLS", line 126
ORA-06512: at line 1

SQL> exec DBMS_RLS.ADD_POLICY ( -
   object_schema       => 'HR', -
   object_name         => 'EMPLOYEES', -
   policy_name         => 'POL_TIME', -
   function_schema     => 'SEC', -
   policy_function     => 'PREDICATE', -
   statement_types     => 'SELECT', -
   policy_type         =>  DBMS_RLS.STATIC)
> > > > > > >
PL/SQL procedure successfully completed.

SQL>
```

2. Create the function used by the security policy to return a predicate. If the user executes the query on the `HR.EMPLOYEES` table after a certain authorized time, the query returns only the rows where the `EMAIL` matches the session username, else it returns all rows whose `SALARY` is less than 3100. **Adapt the time to an appropriate time in the function according to the current time so that the test becomes relevant.**

```
SQL> !date
Thu Apr 25 10:29:28 UTC 2013
SQL> create or replace function PREDICATE
           (obj_schema varchar2, obj_name varchar2)
        return varchar2 is d_predicate varchar2(2000);
begin
```

```
 if to_char(sysdate, 'HH24') >= '10'
    and to_char(sysdate, 'MI')<'35'
    then
     d_predicate := 'email = sys_context (''USERENV'' ,
''SESSION_USER'')';
     else d_predicate := 'salary <= 3100';
     end if;
     return d_predicate;
     end predicate;
/
  2    3    4    5    6    7    8    9   10   11   12   13
Function created.

SQL>
```

3. Connect as SKING to test the VPD policy.

```
SQL> connect sking
Enter password: ******
Connected.
SQL> SELECT email, last_name, salary FROM hr.employees;

EMAIL                     LAST_NAME                     SALARY
------------------------- ------------------------- ----------
SKING                     King                           24000

SQL> !date
Thu Apr 25 10:30:47 UTC 2013

SQL>
```

4. Test under another user.

```
SQL> connect pfay
Enter password: ******
Connected.
SQL> !date
Thu Apr 25 10:36:43 UTC 2013

SQL> SELECT email, last_name, salary FROM hr.employees;

EMAIL                     LAST_NAME                     SALARY
------------------------- ------------------------- ----------
PFAY                      Fay                             6000

SQL>
```

The condition in the function is no more true: nevertheless, it is not reparsed nor reexecuted.

5. You have to change the type of the policy to `DYNAMIC`.

```
SQL> conn sec
Enter password: ********
Connected.
SQL> exec DBMS_RLS.DROP_POLICY ('HR', 'EMPLOYEES','POL_TIME');

PL/SQL procedure successfully completed.

SQL> exec DBMS_RLS.ADD_POLICY ( -
   object_schema       => 'HR', -
   object_name         => 'EMPLOYEES', -
   policy_name         => 'POL_TIME', -
   function_schema     => 'SEC', -
   policy_function     => 'PREDICATE', -
   statement_types     => 'SELECT', -
   policy_type         =>  DBMS_RLS.DYNAMIC)
> > > > > > >
PL/SQL procedure successfully completed.

SQL>
```

6. Recreate the function with an appropriate time.

```
SQL> !date
Thu Apr 25 10:40:54 UTC 2013
SQL> create or replace function PREDICATE
           (obj_schema varchar2, obj_name varchar2)
        return varchar2 is d_predicate varchar2(2000);
begin
 if to_char(sysdate, 'HH24') >= '10'
    and to_char(sysdate, 'MI')<'45'
    then
     d_predicate := 'email = sys_context (''USERENV'' ,
''SESSION_USER'')';
     else d_predicate := 'salary <= 3100';
     end if;
     return d_predicate;
     end predicate;
/

  2    3    4    5    6    7    8    9   10   11   12   13
Function created.
```

```
SQL>
```

7. Connect as SKING and then as PFAY to test the VPD policy.

```
SQL> connect sking
Enter password: ******
Connected.
SQL> SELECT email, last_name, salary FROM hr.employees;

EMAIL                     LAST_NAME                     SALARY
------------------------- ------------------------- ----------
SKING                     King                           24000

SQL> connect pfay
Enter password: ******
Connected.
SQL> /

EMAIL                     LAST_NAME                     SALARY
------------------------- ------------------------- ----------
PFAY                      Fay                             6000

SQL>
```

8. Wait 5 minutes and retest to verify that the function is reexecuted.

```
SQL> !date
Thu Apr 25 10:45:48 UTC 2013
SQL> SELECT email, last_name, salary FROM hr.employees;

EMAIL                     LAST_NAME                SALARY
------------------------- ---------------- -- ----------
AKHOO                     Khoo                       3100
CDAVIES                   Davies                     3100
JFLEAUR                   Fleaur                     3100
ACABRIO                   Cabrio                     3000
AWALSH                    Walsh                      3100
KFEENEY                   Feeney                     3000

6 rows selected.

SQL>
SQL> connect sking
Enter password: ******
Connected.
SQL> SELECT email, last_name, salary FROM hr.employees;
```

```
EMAIL                     LAST_NAME                SALARY
------------------------- ---------------- -- ----------
AKHOO                     Khoo                       3100
CDAVIES                   Davies                     3100
JFLEAUR                   Fleaur                     3100
ACABRIO                   Cabrio                     3000
AWALSH                    Walsh                      3100
KFEENEY                   Feeney                     3000

6 rows selected.

SQL>
```

9. Clean up the `POL_TIME` policy.

```
SQL> connect sec
Enter password: ******
Connected.
SQL> exec DBMS_RLS.DROP_POLICY ('HR', 'EMPLOYEES','POL_TIME')

PL/SQL procedure successfully completed.

SQL>
```

10. Drop the `EMP_USER` context, the `CURRENT_EMP` package and the logon trigger.

```
SQL> DROP CONTEXT EMP_USER;

Context dropped.

SQL> DROP PACKAGE sec.CURRENT_EMP;

Package dropped.

SQL> DROP TRIGGER sec.EMP_LOGON;

Trigger dropped.

SQL>
```

Practice 12-3: Troubleshooting VPD Policies

Overview

In this practice, you will diagnose and troubleshoot VPD policies at creation or execution time.

Tasks

1. Create a VPD policy using the `FUN` function as follows:
   a. Create the function.

```
SQL> create or replace function fun
           (object_schema varchar2, object_name varchar2)
   return varchar2
   is
     d_predicate varchar2(2000);
   BEGIN
     d_predicate :=  '(mail = sys_context (''USERENV'',
''SESSION_USER'')';
     RETURN d_predicate;
    END fun;
/
  2    3    4    5    6    7    8    9   10
Function created.

SQL>
```

   b. Create the VPD policy.

```
SQL> EXEC dbms_rls.drop_policy('HR', 'EMPLOYEES','FUN_POLICY')
BEGIN dbms_rls.drop_policy('HR', 'EMPLOYEES', 'FUN_POLICY')
END;

*
ERROR at line 1:
ORA-28102: policy does not exist
ORA-06512: at "SYS.DBMS_RLS", line 126
ORA-06512: at line 1

SQL> BEGIN
    dbms_rls.add_policy
     (object_schema   => 'HR',  object_name => 'EMPLOYEES',
      policy_name         => 'fun_policy',
      function_schema     => 'SEC',
      policy_function     => 'FUN',
      statement_types     => 'select, index',
      policy_type         => dbms_rls.CONTEXT_SENSITIVE);
```

```
     END;
/
  2    3    4    5    6    7    8    9   10   11
PL/SQL procedure successfully completed.

SQL>
```

2. Connect as `SKING` to test the policy.

```
SQL> conn sking
Enter password: ******
Connected.
SQL> SELECT email FROM hr.employees;
SELECT email FROM hr.employees
                 *
ERROR at line 1:
ORA-28113: policy predicate has error

SQL>
```

You did not get an error at the policy creation but at run time.

3. Trace the statement and analyze the trace file.
   a. Trace your session and reexecute the statement.

```
SQL> ALTER SESSION SET EVENTS '10730 TRACE NAME CONTEXT FOREVER,
LEVEL 1';

Session altered.
SQL> SELECT email FROM hr.employees;
SELECT email FROM hr.employees
                 *
ERROR at line 1:
ORA-28113: policy predicate has error

SQL> EXIT
$
```

   b. Analyze the trace file.

```
$ ls -ltr *ora*.trc
…
lines deleted
…
-rw-r----- 1 oracle oinstall 6083 Apr 25 11:46
orcl_mmon_6671.trc
-rw-r----- 1 oracle oinstall  119 Apr 25 11:49
orcl_ora_21114.trm
```

```
-rw-r----- 1 oracle oinstall 3251 Apr 25 11:49
orcl_ora_21114.trc
$ cat orcl_ora_21114.trc
...
---------------------------------------------------------------
Error information for ORA-28113:
Logon user     : SKING
Table/View     : HR.EMPLOYEES
VPD Policy name    : FUN_POLICY
Policy function: SEC.FUN
RLS view  :
SELECT
"EMPLOYEE_ID","FIRST_NAME","LAST_NAME","EMAIL","PHONE_NUMBER","H
IRE_DATE","JOB_ID","SALARY","COMMISSION_PCT","MANAGER_ID","DEPAR
TMENT_ID" FROM "HR"."EMPLOYEES"   "EMPLOYEES" WHERE ((mail =
sys_context ('USERENV', 'SESSION_USER'))
ORA-00907: missing right parenthesis
---------------------------------------------------------------
$
```

4. Rewrite the function adding the missing right parenthesis in the `d_predicate := '(mail = sys_context (''USERENV'', ''SESSION_USER''))';`.

```
$ sqlplus sec
Enter password: ********
Connected.
SQL> create or replace function fun
           (object_schema varchar2, object_name varchar2)
   return varchar2
  IS
    d_predicate varchar2(2000);
  BEGIN
    d_predicate :=  '(mail = sys_context (''USERENV'',
''SESSION_USER''))';
    RETURN d_predicate;
  END fun;
/
  2    3    4    5    6    7    8    9   10
Function created.

SQL>
```

5. Connect as `SKING` to retest the policy.

```
SQL> conn sking
Enter password: ********
Connected.
```

```
SQL> SELECT mail FROM hr.employees;
SELECT mail FROM hr.employees
                 *
ERROR at line 1:
ORA-28113: policy predicate has error

SQL>
```

6. There is still an error. Proceed as in the previous steps.
   a. Trace your session and reexecute the statement.

```
SQL> ALTER SESSION SET EVENTS '10730 TRACE NAME CONTEXT FOREVER,
LEVEL 1';

Session altered.
SQL> SELECT mail FROM hr.employees;
SELECT mail FROM hr.employees
                 *
ERROR at line 1:
ORA-28113: policy predicate has error

SQL> EXIT
$
```

   b. Analyze the trace file.

```
$ cd $ORACLE_BASE/diag/rdbms/orcl/orcl/trace
$ ls -ltr *ora*.trc
…
lines deleted
…
-rw-r----- 1 oracle oinstall 6345 Apr 25 11:56
orcl_mmon_6671.trc
-rw-r----- 1 oracle oinstall  100 Apr 25 11:59
orcl_ora_22796.trm
-rw-r----- 1 oracle oinstall 3258 Apr 25 11:59
orcl_ora_22796.trc
$ cat orcl_ora_22796.trc
...
```

```
*** 2013-04-25 11:59:04.588
------------------------------------------------------------
------------------------------------------------------------
Error information for ORA-28113:
Logon user     : SKING
Table/View     : HR.EMPLOYEES
VPD Policy name    : FUN_POLICY
```

```
Policy function: SEC.FUN
RLS view  :
SELECT
"EMPLOYEE_ID","FIRST_NAME","LAST_NAME","EMAIL","PHONE_NUMBER","H
IRE_DATE","JOB_ID","SALARY","COMMISSION_PCT","MANAGER_ID","DEPAR
TMENT_ID" FROM "HR"."EMPLOYEES"   "EMPLOYEES" WHERE ((mail =
sys_context ('USERENV', 'SESSION_USER')))
ORA-00904: "MAIL": invalid identifier
---------------------------------------------------------------
$
```

7. Rewrite the function with the right column name: `EMAIL`.

```
$ sqlplus sec
Enter password: ********
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> create or replace function fun
           (object_schema varchar2, object_name varchar2)
  return varchar2
  IS
    d_predicate varchar2(2000);
  BEGIN
    d_predicate :=  '(email = sys_context (''USERENV'',
''SESSION_USER''))';
    RETURN d_predicate;
  END fun;
/
  2    3    4    5    6    7    8    9   10
Function created.

SQL>
```

8. Connect as `SKING` to retest the policy.

```
SQL> conn sking
Enter password: ********
Connected.
SQL> SELECT email FROM hr.employees;

EMAIL
-------------------------
SKING
```

```
SQL>
```

Practice 12-4: Cleaning Up VPD Policies

Overview

In this practice, you will drop all VPD policies.

Tasks

1. Find all VPD policies.

```
SQL> conn sec
Enter password: ******
Connected.
SQL> SELECT policy_name FROM dba_policies;

POLICY_NAME
-------------------------------------------------------------
… rows deleted …
FUN_POLICY
… rows deleted …

SQL>
```

2. Drop each VPD policy listed in step 1.

```
SQL> exec DBMS_RLS.DROP_POLICY ('HR','EMPLOYEES','FUN_POLICY')

PL/SQL procedure successfully completed.

SQL> EXIT
$
```