SWISS ORACLE USER GROUP

SOUG www.soug.ch

Newsletter 5/2014

Sonderausgabe

- OBIF
- DB licensing with VMware
- Delphix
- 12c:
  SQL Plan / Security Features

Miguel Anjo, Trivadis

7 significant changes to user access control and privileges in Oracle 12.1

Oracle 12c is bringing several new ways to ensure the security of the databases we manage. This short article describes seven changes related to users' privileges and access that appeared in releases 12.1.0.1 and 12.1.0.2.

1. READ and READ ANY TABLE privileges

Since long there was **SELECT, INSERT, UPDATE** and **DELETE** privileges over tables. With Oracle 12.1.0.2 there is the new **READ** and **READ ANY TABLE** privilege. The reason is that with the **SELECT** privilege the user is able to do **"LOCK TABLE xxx IN EXCLUSIVE MODE"** and **"SELECT … FOR UPDATE"** and therefore locking table for other users.

For this reason, once you migrate to Oracle 12c, it might be interesting to replace the **SELECT** privileges with **READ** privileges, like on read only users or roles.

Care should be taken for some strict situations as the READ privilege is not compliant with SQL92 security standard. Within this norm, an UPDATE or DELETE privilege requires in addition the SELECT privilege in order to perform DDL operations. Databases running Oracle Database Vault or having the SQL92_SECURITY parameter set to TRUE (the default is FALSE) should take particular attention.

2. Attach roles to PL/SQL program units – code based access control

The feature mentioned in this paragraph could give a full length article. Let's try to make it simple. As of version 12.1 it is possible to assign roles to PL/SQL code like functions, procedures and packages. This gives the possibility of raising user privileges only during the run time of the program, which is a big advantage for running code with invokers' rights (**authid current_user**). Until Oracle 11g for achieving the same there were two options: to grant the privileges directly to the user executing the procedure; or to run with definer rights – which in turn could be much more powerful than the necessary – and it would lose the caller environment information.

Let's see an example of how useful is this new feature. Imagine user **BERN** writes the following procedure to be used by other users:

```
CREATE OR REPLACE PROCEDURE bern.proc_current_user
  AUTHID CURRENT_USER
  AS
  BEGIN
    INSERT INTO bern.table1 (user, date)
      VALUES (SYS_CONTEXT('USERENV', 'CURRENT_USER'));
    COMMIT;
  END;
/

GRANT EXECUTE ON bern.proc_current_user TO geneva;

CREATE ROLE r_insert;
GRANT INSERT ON bern.table1 TO r_insert;
```

Until Oracle 11g the user **BERN** needs to give the following permissions:

```
GRANT r_insert TO geneva;
```

This means that as soon as user **GENEVA** logins, he has full insert privileges on **bern.table1** which is not secure in most situations.

From Oracle 12c it is now possible to user **BERN** to pass the privileges only to the procedure:

```
GRANT r_insert TO PROCEDURE bern.proc_current_user;
```

In order to help visualize the security context during a session within the different options of the example one can look at the following diagram.

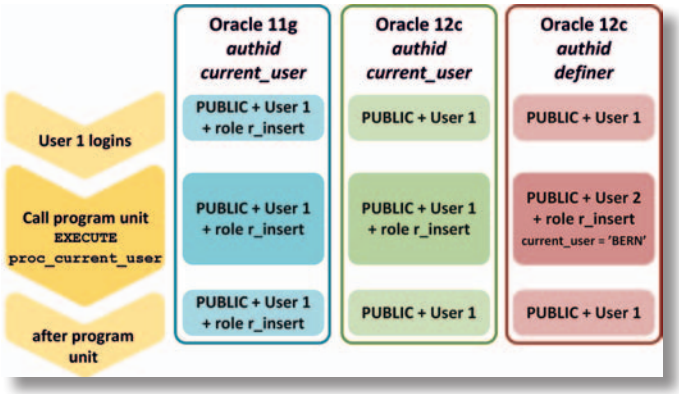

Figure 1 – Security context changes during call to program unit

It is visible that by using this new feature it is possible to keep to a minimum the privileges assigned to a user in a permanent fashion while not losing on functionality.

Oracle's idea behind this new possibility is to segregate the roles between the person who develops the code, the one who deploys the application and a privilege manager. It is the privilege manager the responsible for creating the necessary roles and granting them to the piece of code.

3. DELEGATE OPTION for roles

In order to assign existing roles to program units, a user needs to have GRANT ANY ROLE privilege or the ADMIN option for role. Since 12.1.0.2 a new grant option exists – DELEGATE – specifically for code based access control. As an example, privilege manager user ZURICH creates a role and grants to BERN which should then be able to grant it to a program unit:

```
CREATE ROLE r_insert;
GRANT SELECT, UPDATE ON bern.table1 TO r_insert;
GRANT r_insert TO bern WITH DELEGATE OPTION;
```

Then BERN has now the right to delegate the role to a program unit:

```
GRANT r_insert TO PROCEDURE bern.proc_current_user;
```

4. Proxy only connect user property

This yet undocumented feature allows to define application schemas which can only be accessed through a proxy user. It makes a very useful to assure that no user connects directly to the application schema, even by knowing its password.

Here how it works:

```
SQL> CREATE USER app_user IDENTIFIED BY xyz;
User created.

SQL> GRANT CREATE SESSION TO app_user;
Grant succeeded.

SQL> ALTER USER app_user PROXY ONLY CONNECT;
User altered.

SQL> CREATE USER personal_user IDENTIFIED BY prx1;
User created.

SQL> ALTER USER app_user GRANT CONNECT THROUGH personal_user;
User altered.

SQL> CONNECT app_user/xyz;
ERROR:
ORA-28058: login is allowed only through a proxy

SQL> CONNECT personal_user[app_user]/prx1;
Connected.

SQL> SELECT user FROM dual;
USER
------------------------------
APP_USER
```

As usual, the use of undocumented features are not supported by Oracle. The syntax to rollback the change is:

```
SQL> ALTER USER app_user CANCEL PROXY ONLY CONNECT;
```

5. RESOURCE role without UNLIMITED TABLESPACE privilege

While it is not recommended by Oracle to continue to use the RESOURCE role ("this role might not be created automatically by future releases of Oracle Database" as is written in the documentation), there is an important change on Oracle 12.1 – there is no more the UNLIMITED TABLESPACE system privilege.

The consequence of this change is that, when you grant the RESOURCE role, you need also to explicitly grant some quota on a tablespace to the user.

Imagine the common (not recommended) way to create users and a table:

```
SQL> CREATE USER bern IDENTIFIED BY xyz;
SQL> GRANT CONNECT, RESOURCE TO bern;

SQL> CONNECT bern/xyz
SQL> CREATE TABLE t1(c1 NUMBER);
```

Up to 11.2.0.4 you could do after as user 'BERN':

```
SQL> INSERT INTO t1 VALUES (1);
1 row created.
```

However from 12.1 this is what happens:

```
SQL> INSERT INTO t1 VALUES (1);
INSERT INTO t1 VALUES (1)

ERROR at line 1:
ORA-01950: no privileges on tablespace 'USERS'
```

6. SELECT ANY DICTIONARY protects critical SYS schema tables

As DBA we know that hashed passwords can be found at **USER$** table. Some other tables like **DEFAULT_PWD$, ENC$, LINK$, USER_HISTORY$, CDB_LOCAL_ADMINAUTH$** and **XS$VERIFIERS** also contain very sensitive information.

Starting with Oracle 12c, the **SELECT ANY DICTIONARY** role does not include access to those views, limiting only to SYS user the access to them.

7. Last login information

Another small improvement that can greatly help to keep an environment safe is the information about the last successful login. It permits, without any other auditing, to know when an account was last used. This allows to make the DBA sure when he can delete an unused account and also can make the user aware if someone else abused of his account.

This information is visible at the new **LAST_LOGIN** column of the **DBA_USERS** view.

Logging in to SQL*Plus the user can also see when the last successful login happen:

```
$sqlplus bern/xyz

SQL*Plus: Release 12.1.0.1.0 Production on Thu Oct 30 12:38:25 2014
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Last Successful login time: Thu Oct 30 2014 12:28:32 +01:00
```


Contact

Trivadis

Miguel Anjo
E-Mail:
miguel.anjo@trivadis.com