CHAPTER 18

Basic Oracle Security

Basic Oracle Security

- Information is vital to success, but when damaged or in the wrong hands, it can threaten success
- Oracle provides extensive security features to safeguard your information from both unauthorized viewing and intentional or inadvertent (unintentional) damage
- This security is provided by granting or revoking privileges on a person-by-person and Privilege by-privilege basis
- Oracle uses the create user, create role, and grant commands to control data access

Basic Oracle Security

- In this chapter, you will see the basic security features available in Oracle

Users, Roles, and Privileges

- Every Oracle user has a name and password and owns any tables, views, and other resources that he or she creates
- An Oracle role is a set of privileges (or the type of access that each user needs, depending on his or her status and responsibilities)
- You can grant or bestow (give) specific privileges to roles and then assign roles to the appropriate users
- A user can also grant privileges directly to other users.

Users, Roles, and Privileges

- Database system privileges let you execute specific sets of commands
- The CREATE TABLE privilege, for example, lets you create tables
- The privilege GRANT ANY PRIVILEGE allows you to grant any system privilege

Users, Roles, and Privileges

- Database object privileges give you the ability to perform some operation on various objects
- The DELETE privilege, for example, lets you delete rows from tables and views
- The SELECT privilege allows you to query with a select from tables, views, sequences, and snapshots (materialized views)

Users, Roles, and Privileges

- See “Privilege” in the Alphabetical Reference at the end of this book for a complete list of system and object privileges

System privileges

- There are quite a few system privileges: in Oracle 9.2, we count 157 of them, and 10g has even 166. Those can be displayed with
  - select name from system_privilege_map;
- Executing this statement, we find privileges like *create session, drop user, alter database*

Object privileges

- privileges can be assigned to the following types of database objects:
  - Tables
    select, insert, update, delete, alter, debug, flashback, on commit refresh, query rewrite, references, all
  - Views
    select, insert, update, delete, under, references, flashback, debug

Creating a User

- The Oracle system comes with many users already created, including SYSTEM and SYS
- The SYS user owns the core internal tables Oracle uses to manage the database;
- SYSTEM owns additional tables and views
- You can log onto the SYSTEM user to create other users because SYSTEM has that privilege.

Creating a User

- When installing Oracle, you (or a system administrator) first create a user for yourself
- This is the simplest format for the **create user** command:

```
create user user identified
{by password | externally | globally as 'extnm'};
```

- Many other account characteristics can be set via this command;
- see the **create user** command in the Alphabetical Reference for details

Creating a User

- A system administrator (who has a great many privileges) may want the extra security of having a separate password
- Let's call the system administrator Dora in the following examples:

```
create user Dora identified by avocado;
```

Creating a User

- Dora's account now exists and is secured by a password
- You also can set up the user with specific tablespaces and quotas (limits) for space and resource usage
- See **create user** in the Alphabetical Reference and Chapters 17 and 20 for a discussion of tablespaces and resources. Or

https://www.questionmark.com/perception/help/rms/v1/manuals/ug/Content/installing/Installation/database/Creating_an_Oracle_Tablespace_and_User.htm#TB10

Creating a User

- To change a password, use the **alter user** command:

```
alter user Dora identified by psyche;
```

- Now Dora has the password “psyche” instead of “avocado.”
- However, Dora cannot log into her account until she first has the CREATE SESSION system privilege

```
grant CREATE SESSION to Dora;
```

Password Management

- Passwords can expire, and accounts may be locked due to repeated failed attempts to connect
- When you change your password, a password history may be maintained to prevent reuse of previous passwords
- The expiration characteristics of your account's password are determined by the profile assigned to your account

Password Management

- Profiles, which are created by the create profile command, are managed by the DBA (database administrator, discussed later in the chapter)
- See the CREATE PROFILE entry in the Alphabetical Reference for full details on the create profile command

Password Management

- Relative to passwords and account access, profiles can enforce the following:
  - The "lifetime" of your password, which determines how frequently you must change it
  - The grace period following your password's "expiration date" during which you can change the password
  - The number of consecutive failed connect attempts allowed before the account is automatically "locked"

Password Management

- The number of days the account will remain locked
- The number of days that must pass before you can reuse a password
- The number of password changes that must take place before you can reuse a password

- Database administrators can change any user's password via the **password** command;
- other users can change only their own password

Password Management

- When you enter the password command, you will be prompted for the old and new passwords, as shown in the following listing:

```
connect dora/psyche
Password
Changing password for dora
Old password:
New password:
Retype new password:
```

- When the password has been successfully changed, you will receive the feedback:

```
Password changed
```

Password Management

- You can set another user's password from a DBA account
- Simply append the username to the password command
- You will not be asked for the old password:

```
password dora
Changing password for dora
New password:
Retype new password:
```

Enforcing Password Expiration

- You can use profiles to manage the expiration, reuse, and complexity of passwords
- You can limit the lifetime of a password, and lock an account whose password is too old
- You can also force a password to be at least moderately complex, and lock any account that has repeated failed login attempts

Enforcing Password Expiration

- For example, if you set the FAILED_LOGIN_ATTEMPTS resource of the user's profile to 5,
- Then four consecutive failed login attempts will be allowed for the account;
- **The fifth will cause the account to be locked**

Enforcing Password Expiration

- NOTE:
- If the correct password is supplied on the fifth attempt, the "failed login attempt count" is reset to 0,
- allowing for five more consecutive unsuccessful login attempts before the account is locked
- In the following listing, the LIMITED_PROFILE profile is created for use by the user JANE:

Enforcing Password Expiration

```
create profile LIMITED_PROFILE limit
FAILED_LOGIN_ATTEMPTS 5;

create user JANE1 identified by EYRE1
profile LIMITED_PROFILE;

grant CREATE SESSION to JANE1;
```

- If there are five consecutive failed connection attempts to the JANE account, the account will be automatically locked by Oracle
- When you then use the correct password for the JANE account, you will receive an error.

Enforcing Password Expiration

```
create profile LIMITED_PROFILE limit
FAILED_LOGIN_ATTEMPTS 5;

create user JANE identified by EYRE
profile LIMITED_PROFILE;

grant CREATE SESSION to JANE;
```

- If there are five consecutive failed connection attempts to the JANE account, the account willbe automatically locked by Oracle
- When you then use the correct password for the JANE account, you will receive an error.

Enforcing Password Expiration

connect jane1/eyre1

ERROR:

ORA-28000: the account is locked //////19/09/12

- To unlock the account, use the **account unlock** clause of the **alter user** command (from a DBA account), as shown in the following listing:

  alter user JANE account unlock;

- Following the unlocking of the account, connections to the JANE account will once again be allowed

Enforcing Password Expiration

- You can manually lock an account via the **account lock** clause of the **alter user** command

  alter user JANE account lock;

  NOTE

- You can specify account lock as part of the create user command

Enforcing Password Expiration

- If an account becomes locked due to repeated connection failures, it will automatically become unlocked when its profile's PASSWORD_LOCK_TIME value is exceeded
- For example, if PASSWORD_LOCK_TIME is set to 1, the JANE account in the previous example would be locked for one day, after which the account would be unlocked again

Enforcing Password Expiration

- You can establish a maximum lifetime for a password via the PASSWORD_LIFE_TIME resource within profiles
- For example, you could force users of the LIMITED_PROFILE profile to change their passwords every 30 days:

```
alter profile LIMITED_PROFILE limit
PASSWORD_LIFE_TIME 30;
```

Enforcing Password Expiration

- In this example, the **alter profile** command is used to modify the LIMITED_PROFILE profile
- The PASSWORD_LIFE_TIME value is set to 30, so each account that uses that profile will have its password expire after 30 days
- If your password has expired, you must change it the next time you log in, unless the profile has a specified grace period for expired passwords

Enforcing Password Expiration

- The grace period parameter is called PASSWORD_GRACE_TIME
- If the password is not changed within the grace period, the account expires
- NOTE
- If you are going to use the PASSWORD_LIFE_TIME parameter, you need to give the users a way to change their passwords easily

Enforcing Password Expiration

- An "expired" account is different from a "locked" account
- A locked account, as discussed earlier in this section, may be automatically unlocked by the passage of time
- An expired account, however, requires manual intervention by the DBA to be reenabled

Enforcing Password Expiration

- NOTE
- If you use the password expiration features, make sure the accounts that own your applications have different profile settings; otherwise,
- the accounts may become locked and the application may become unusable.

Enforcing Password Expiration

- To reenable an expired account, execute the alter user command
- In this example, the DBA manually expires JANE's password:

```
alter user jane password expire;
```

- Next, JANE attempts to connect to her account
- When she provides her password, she is immediately prompted for a new password for the account

Enforcing Password Expiration

```
connect jane/eyre
ERROR:
ORA-28001: the password has expired
Changing password for jane
New password:
Retype new password:
Password changed
Connected.
```

Enforcing Password Reuse Limitations

- To prevent a password from being reused, you can use one of two profile parameters:
  - PASSWORD_REUSE_MAX or
  - PASSWORD_REUSE_TIME
- These two parameters are mutually exclusive; if you set a value for one of them, the other must be set to UNLIMITED(important)

Enforcing Password Reuse Limitations

- The PASSWORD_REUSE_TIME parameter specifies the number of days that must pass before a password can be reused
- For example, if you set PASSWORD_REUSE_TIME to 60, you cannot reuse the same password within 60 days
- The PASSWORD_REUSE_MAX parameter specifies the number of password changes that must occur before a password can be reused

Enforcing Password Reuse Limitations

- If you attempt to reuse the password before the limit is reached, Oracle will reject your password change
- For example,
  - you can set PASSWORD_REUSE_MAX for the LIMITED_PROFILE profile created earlier in this chapter:

```
alter profile LIMITED_PROFILE limit
PASSWORD_REUSE_MAX 3
PASSWORD_REUSE_TIME UNLIMITED;
```

Enforcing Password Reuse Limitations

- If the user JANE now attempts to reuse a recent password, the password change attempt will fail
- For example, suppose she changes her password, as in the following line:

```
alter user JANE identified by austen;
```

- And then she changes it again:

```
alter user JANE identified by whitley;
```

Enforcing Password Reuse Limitations

- During her next password change, she attempts to reuse a recent password, and the attempt fails:

```
alter user jane identified by austen;
alter user jane identified by austen
*
ERROR at line 1:
ORA-28007: the password cannot be reused
```

- She cannot reuse any of her recent passwords; she will need to come up with a new password.

Standard Roles

- At the beginning of this chapter, we created a user named Dora
- Now that Dora has an account,
- what can she do in Oracle?
- At this point, nothing—Dora has no system privileges other than CREATE SESSION

Standard Roles

- In many cases, application users receive privileges via roles
- You can group system privileges and object accesses into roles specific to your application users' needs
- You can create your own roles for application access, and you can use Oracle's default roles for some system access requirements.

Standard Roles

- The most important standard roles created during database creation are:

CONNECT, RESOURCE, and DBA	Provided for compatibility with previous versions of Oracle database software.
DELETE_CATALOG_ROLE, EXECUTE_CATALOG_ROLE, SELECT_CATALOG_ROLE	These roles are provided for accessing data dictionary views and packages.
EXP_FULL_DATABASE, IMP_FULL_DATABASE	These roles are provided for convenience in using the Import and Export utilities.
AQ_USER_ROLE, AQ_ADMINISTRATOR_ROLE	These roles are needed for Oracle Advanced Queuing.
SNMPAGENT	This role is used by the Enterprise Manager Intelligent Agent.
RECOVERY_CATALOG_OWNER	This role is required for the creation of a recovery catalog schema owner.
SCHEDULER_ADMIN	This role allows the grantee to execute the DBMS_ SCHEDULER package's procedures; should be restricted to DBAs.

Standard Roles

- CONNECT, RESOURCE, and DBA are provided for backward compatibility and should no longer be used
- CONNECT gives users the ability to log in and perform basic functions
- Users with CONNECT may create tables, views, sequences, clusters, synonyms, and database links

Standard Roles

- Users with RESOURCE can create their own tables, sequences, procedures, triggers, datatypes, operators, indextypes, indexes, and clusters
- Users with the RESOURCE role also receive the UNLIMITED TABLESPACE system privilege, allowing them to bypass quotas on all tablespaces

Standard Roles

- Users with the DBA role can perform database administration functions, including creating and altering users, tablespaces, and objects
- In place of CONNECT, RESOURCE, and DBA, you should create your own roles that have privileges to execute specific system privileges

Format for the grant Command

- In the following sections you will see how to grant privileges to users and roles
- format for the **grant** command for system privileges

```
grant {system privilege | role | all [privileges] }
[, {system privilege | role | all [privileges] }. . .]
to {user | role} [, {user | role}]. . .
[identified by password ]
[with admin option]
```

Format for the grant Command

- You can grant any system privilege or role to another user, to another role, or to **public**
- The **with admin option** clause permits the grantee to bestow (give) the privilege or role on other users or roles
- The **all** clause grants the user or role all privileges except the SELECT ANY DICTIONARY system privilege

Revoking Privileges

- Privileges granted can be taken away
- The revoke command is similar to the grant command:

```
revoke {system privilege | role | all [privileges] }
[, {system privilege | role | all [privileges] }. . .]
from {user | role} [, {user | role}]. . .
```

Revoking Privileges

- An individual with the DBA role can revoke CONNECT, RESOURCE, DBA, or any other privilege or role from anyone, including another DBA
- This, of course, is dangerous, and is why DBA privileges should be given neither lightly nor to more than a tiny minority who really need them.

Revoking Privileges

- To remove a user and all the resources owned by that user, use the **drop user** command, like this:

```
drop user username [cascade];
```

- The cascade option drops the user along with all the objects owned by the user, including referential integrity constraints

Revoking Privileges

- The cascade option invalidates views, synonyms, stored procedures, functions, or packages that refer to objects in the dropped user's schema
- If you don't use the cascade option and there are still objects owned by the user,
- Oracle does not drop the user and instead returns an error message

What Users Can Grant

- A user can grant privileges on any object he or she owns
- The database administrator can grant any system privilege.
- Suppose that user Dora owns the COMFORT table and is a database administrator
- Create two new users, Bob and Judy, with these privileges:

What Users Can Grant

create user Judy identified by sarah;

User created.

grant CREATE SESSION to Judy;

Grant succeeded.

create user Bob identified by carolyn;

User created.

grant CREATE SESSION, CREATE TABLE, CREATE VIEW,

CREATE SYNONYM to bob;

Grant succeeded.

alter user bob

default tablespace users

quota 5m on users;

User altered.

What Users Can Grant

- This sequence of commands gives both Judy and Bob the ability to connect to Oracle, and
- gives Bob some extra capabilities
- But can either do anything with Dora's tables? Not without explicit access.
- The privileges a user can grant include these: