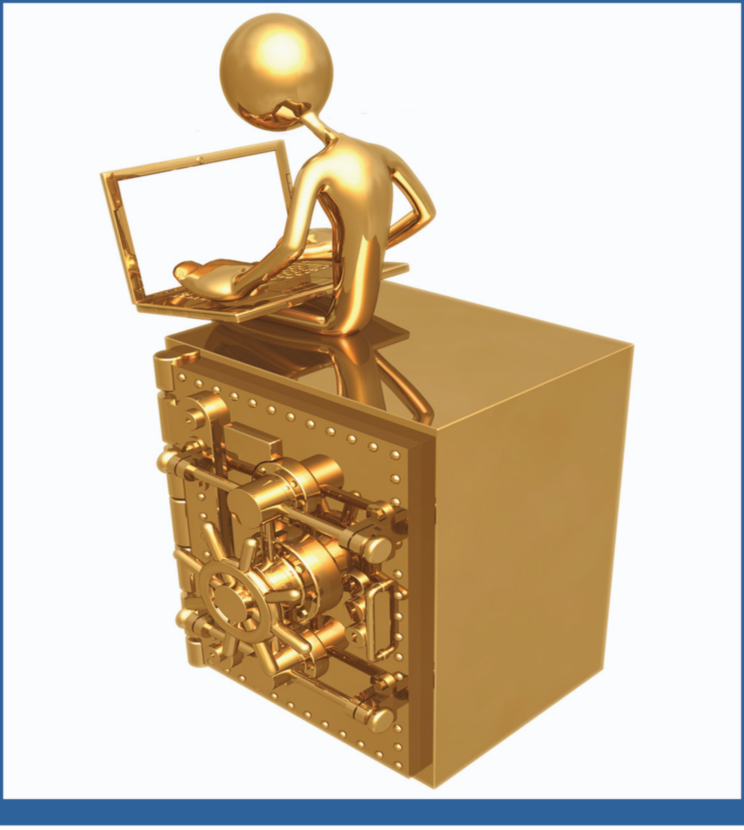
CRC Press
Taylor & Francis Group
AN AUERBACH BOOK

HOWTO Secure and Audit Oracle 10g and 11g



Ron Ben Natan

Foreword by Pete Finnigan

Chapter 4

Account Security

Oracle security is managed through the concept of a database user. To connect to the database you must have a valid Oracle user account. The permissions you have within the database are based on the privileges that are assigned to the user through which you are connected to Oracle. Account security is therefore one of the fundamental building blocks in ensuring a secure Oracle environment.

Oracle user accounts are created and managed within the database. In this chapter you will focus on how to create users, how to manage passwords within Oracle, how to limit accounts, and how to review which limits are assigned to accounts. This chapter does not deal with how the user is authenticated with Oracle. As you'll see in Chapter 6 there are many ways by which a user can be authenticated—some using the Oracle database and some using an external authentication mechanism. Regardless of the method by which you authenticate a user, when the connection is made it is managed as a user account within the database. When a logon to Oracle occurs, a set of credentials is presented to Oracle. Oracle (or an external authenticator that Oracle is working with) determines whether these credentials are correct and whether to allow the logon. After authentication occurs and the logon succeeds, the connection is allowed as the database user—where all of the account properties are derived from.

Although you will not learn about authentication in this chapter, you will learn quite a number of things about passwords. Weak and default passwords are considered by many to be the number one reason for security breaches in database environments. Older Oracle installations in which the change_on_install password was never changed exist even today and instances where the default password for the DBSNMP account has not been changed are quite common. You'll see how to combat this type of misuse as well as how to manage password profiles in general.

4.1 HOWTO Create, Alter, Drop, and Lock User Accounts

Use the CREATE USER command to create an account:

```
SQL> create user ronbn identified by "0okmnhy^";
User created.
```

Use a strong password when you create users and force your users to change their passwords (either by creating the account with an expired password or through password profiles which we'll learn about in the next HOWTO). Once the user has been created, grant it privileges depending on what this user needs to do. If the user needs to be able to log onto the database (most any user will need that) then at a minimum you need to allow them to create a session. If you don't the user will get the following error when trying to logon:

```
SQL> connect ronbn
Enter password: ********
ERROR:
ORA-01045: user RONBN lacks CREATE SESSION privilege; logon
denied
```

Grant the privilege and then the user can logon:

```
SQL> grant create session to ronbn;
Grant succeeded.
```

If you want to ensure that the user changes their password the first time they connect, use the PASSWORD EXPIRE option:

```
SQL> create user ronbn identified by "temp8uhbgt5" password expire;
User created.
SQL> grant create session to ronbn;
Grant succeeded.
SQL> connect ronbn
Enter password: ***********
ERROR:
ORA-28001: the password has expired
Changing password for ronbn
New password: ********
Retype new password: ********
Password changed
Connected.
```

Now that ronbn can connect to the database, grant other privileges as required but use the general concept of minimum privileges—i.e., grant privileges on an as-needed basis. As an example, assume that ronbn will be used within an application and needs to be able to create tables within the schema. Go ahead and grant them CREATE TABLE privileges:

```
SQL> grant create table to ronbn;
Grant succeeded.
```

From a privilege perspective, ronbn can create tables within the ronbn schema—but this is not enough. One of the important things most users will need is a quota on a tablespace. Each user is assigned to a tablespace (or multiple tablespaces). Most users will need to create database objects in which case it is not enough for them to have a tablespace, they also need to have a quota on the tablespace. As an example, if you log on as ronbn you can select data but you still cannot create a table:

```
SQL> select count(*) from user_tables;

  COUNT(*)
----------
         0

1 row selected.
```

```
SQL> create table my_first_app_table(i int, j int);
create table my_first_app_table(i int, j int)

*

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

As the error indicates, the insufficient privileges error does not stem from the inability to use the create table command (you just granted this privilege)—it is the lack of quota on the tablespace. To give the user quota on the tablespace use:

```
SQL> alter user ronbn quota 10M on users;
User altered.
```

At this point ronbn can create the table:

```
SQL> connect ronbn
Enter password: ********
Connected.
SQL> create table my_first_app_table(i int, j int);
Table created.
```

You can check your quotas when logged in as database administrator (DBA):

```
SQL> select username, tablespace_name, max_bytes

  2 from dba_ts_quotas
  3 order by username;

USERNAME                         TABLESPACE_NAME                  MAX_BYTES
-------------------------------  ------------------------------  ----------

DMSYS                            SYSAUX                           209715200
OLAPSYS                          SYSAUX                                  -1
RONBN                            USERS                             10485760
SCOTT                            SYSTEM                                  -1
SYSMAN                           SYSAUX                                  -1
TEST1                            SYSTEM                             1048576
TEST1                            USERS                              1048576

7 rows selected.
```

Quotas are generally viewed as attributes related to data management but they should also be viewed as a security mechanism and the grant of a quota is actually managed as a privilege. The reason is that large quotas can be misused to create a denial-of-service (DoS) attack. For example, if an attacker manages to connect as a user that has an unlimited quota or a user with a very large quota they can start filling tables with random data until the disk runs out or until the database's quota at the operating system level is exceeded—at which point the database will stop functioning. Never assign the UNLIMITED TABLESPACE system privilege to anyone but a DBA and continuously audit which users have this (UNLIMITED) system privilege. If you want to revoke the ability to create objects in a tablespace simply alter the user:

```
SQL> alter user ronbn quota 0 on users;
User altered.
```

At which point the user will get the following error:

```
SQL> create table my_second_app_table(i int, j int);

*

ERROR at line 1:
ORA-01536: space quota exceeded for tablespace 'USERS'
```

Of course, everything you just saw is part of the CREATE USER command—you can create a user and all the associated attributes in one go:

```
SQL> create user ronbn
  2 identified by "0okmnhy^"
  3 default tablespace users
  4 quota 10M on users
  5 profile app _ user;

User created.

SQL> grant create session to ronbn;
Grant succeeded.
```

Note that to run these commands you need to create the password profile before you make this call—that's the topic of the next HOWTO.

The ALTER USER format is similar:

```
SQL> alter user ronbn
  2 identified by "p455w0rd"
  3 default tablespace users
  4 quota 20M on users
  5 profile app_user;
User altered.
```

Each of the attributes themselves can be changed with the ALTER USER command. The most commonly used option is to change the password:

```
SQL> alter user ronbn identified by "0okmnhy6";
User altered.
```

Use this command if you are a DBA changing the password for another user or if you are changing your own password:

```
SQL> -- connected as system
SQL> alter user ronbn identified by "0okmnhy6";
User altered.

SQL> connect ronbn/0okmnhy6;
Connected.
SQL> alter user ronbn identified by "9ijnbgt5";
User altered.
```

HOWEVER—you should ALWAYS use the PASSWORD command rather than ALTER USER. You'll see why in Chapter 7 but for curiosity's sake, it is because ALTER USER passes the password in cleartext over the wire whereas PASSWORD passes just the hash value.

There is a difference between this process when you're changing your own password and when you're changing someone else's password (if you have the right privileges). If you're a DBA changing someone else's password you are not asked for the old password (one of the cases when you would do this would be when the user forgot their password). If you are changing your own password you need to type in the old password first so that someone doesn't just walk by your open session when you're away from your desk and change your password (regardless—don't leave sessions open and walk away). You should prefer to use PASSWORD than the ALTER USER command because the password is never visible on the screen (ALTER USER is useful in noninteractive password changes—e.g., within scripts):

```
SQL> connect system
Enter password: ********
Connected.
SQL> password ronbn
Changing password for ronbn
New password: ********
Retype new password: ********
Password changed
SQL> connect ronbn
Enter password: ********
Connected.
SQL> password ronbn
Changing password for ronbn
Old password: ********
New password: *****
Retype new password: *****
Password changed
```

To delete an account use the DROP USER command:

```
SQL> drop user ronbn cascade;
User dropped.
```

When you drop a user with the CASCADE option all of the user's schema objects are dropped and the schema is deleted from the data dictionary. Using CASCADE also ensures that all foreign keys and other dependencies on objects contained within the user schema are dropped.

A user cannot be dropped if there is an active connection using that user account. If you try to drop ronbn while there is a session open you will get the following error:

```
SQL> drop user ronbn cascade;
drop user ronbn cascade

*

ERROR at line 1:
ORA-01940: cannot drop a user that is currently connected
```

To drop the user you will have to find all sessions currently open for that user, kill them, and only then drop the user. Use V$SESSION to find all the SID-s and SERIAL#-s for that user:

```
SQL> select sid,serial#,username from v$session where username='RONBN';
```

```
          SID      SERIAL# USERNAME
---------------  ------------  --------------------------

          145          784  RONBN
          156          633  RONBN

2 rows selected.
```

Next, use the KILL command to terminate these sessions:

```
SQL> alter system kill session '145,784';
System altered.
SQL> alter system kill session '156,633';
System altered.
```

Finally, drop the user:

```
SQL> drop user ronbn cascade;
User dropped.
```

If you don't want to drop the user but want to prevent them from connecting to the database you can lock their account and expire their password. These are both done using the ALTER USER command. To lock an account:

```
SQL> alter user ronbn account lock;
User altered.
```

At which point if the user tries to logon they get the following error message:

```
SQL*Plus: Release 10.2.0.1.0 - Production on Sun Jan 20 13:17:54 2008
Copyright (c) 1982, 2005, Oracle. All rights reserved.
ERROR:
ORA-28000: the account is locked
```

To unlock the account:

```
SQL> alter user ronbn account unlock;
User altered.
```

If you want to make sure that the user will need to change their password in case you later unlock their account then:

```
SQL> alter user ronbn account lock password expire;
User altered.
```

Two Things to Remember about Actions on Accounts

1. Use PASSWORD EXPIRE to ensure that the user has to change their passwords.
2. Always change password using the PASSWORD command and not the ALTER USER command.

4.2 HOWTO Understand the Standard Logon Process

Passwords are the most common methods in which Oracle authenticates you when you logon to the database. Ensuring that users use strong passwords is one of the major things that you can do to protect your database. There are many things that Oracle does by default to ensure that passwords are not compromised and that the normal authentication schema is secure—but there are also things that fall within your realm of responsibility. Passwords need to be secured throughout their lifetimes. They need to be complex enough so as not to be easily guessed. They need to be secured when they travel on the network. The authentication process needs to prevent a brute-force attack in which passwords are guessed endlessly, etc.

To log you onto the database Oracle needs to get a password from you to compare it to something it has previously stored. When a client makes a connection Oracle needs to take the password that you type in when you logon and send something to the database server. The server then compares that to something already stored within the server. Passwords in Oracle are never sent in the clear and are never stored in the clear. In Oracle 11g passwords are encrypted using advanced encryption standard (AES) before they are sent over the network—irrespective of whether or not you enable network encryption. In Oracle 10g the logon scheme does not use AES but rather uses a proprietary scheme called O3LOGON. This scheme ensures that the password cannot be compromised if someone is sniffing the communication or intercepts it. The scheme works as follows:

1. The client sends the logon name to the database (a hex dump from part of a packet is shown in Listing 4.1).
2. The database sends a challenge created by encrypting a random number with the user's password hash (Listing 4.2). The random key can be thought of as session key and is different for each session.
3. The client computes the hash from the password and decrypts the random number—so now the client and the server have a common session key.
4. The client uses this random number as a key to encrypt the password and sends it to the database (Listing 4.3).

When the password is stored in USER$, it is not stored in the clear—instead, a hash value is maintained. The hash value is based on a one-way hash function—i.e., a function that computes a value that is truly well distributed over all combinations and one which ensures that it is computationally

```
..........v...........
scott...AUTH_TERMINAL...
SERVER2003....AUTH_PROGR
AM_NM...sqlplus.exe....A
```

Listing 4.1 Extract from a logon packet—Client sends the logon name to the database.

```
AUTH_SESSKEY.@@0436D43AF
F246F6B1AAAAA4E9C783DC42
2F69564CD3947CC399CBF9FA
B0ECAF1....AUTH_VFR_DATA
```

Listing 4.2 Extract from a logon packet—Database encrypts a random number with the stored password hash.

```
.........scott.. .....AUTH_PASSWO
RD.@@C4AD8FECF2FEB66BBA3935874F6
81B868C4125985E1E62C4439CD1FB5AB
```

Listing 4.3 Extract from a logon packet—Client sends the encrypted password to the server.

not feasible to compute the original password from the hash value. In Oracle 11g the hash value is based on secure hash algorithm-1 (SHA-1)—considered one of the best one-way functions today. The algorithm in 11g has these steps:

1. Generate random bits (called salt)
2. Concatenate the password with the salt (keep the password case sensitive by default)
3. Compute a hash value using SHA-1
4. Concatenate the hash value with the randomly selected salt—this is what is stored in the database

In step 4, the salt has to be stored because when the user authenticates with the database, the same random bits must be used or the hash will come out completely different.

Prior to 11g the hash algorithm is an Oracle proprietary algorithm that works as follows:

1. Concatenate the username with the password and convert it to uppercase (which is why before Oracle 11g passwords were case insensitive).
2. Convert the string to double-byte characters and pad with zeros until the string's length is a multiple of 8 bytes.
3. Encrypt the string using triple data encryption standard with cipher block coding (3DES)/(CBC) using a fixed key (don't be alarmed—we're not done yet—it's ok to use a fixed key in this step even if the whole world knows what this key is).
4. Take the last 8 bytes of the encrypted text as a new key. This is why it is ok—the first encryption is used to ensure that the last 8 bytes are dependent on the entire string.
5. Encrypt the string again using 3DES/CBC and the new key generated in step 4.
6. Take the last 8 bytes of this string and convert it to printable hex—so you get a hash for 16 printable characters.

The move in Oracle 11g to standard algorithms (AES and SHA-1) is important. Although no one has broken the Oracle proprietary algorithms there is no reason to invent encryption and hashing schemes—it is more comforting to use an industry standard that has gone through VERY rigorous mathematical analysis and is trusted by even the most security-demanding organizations.

Three Things to Remember about the Standard Logon Process

1. Passwords are never sent in cleartext over the wire as part of the logon process—they are always encrypted.
2. The password hash is used to encrypt random session keys. Because the password hash is known to both client and server they are used to enable the sharing of these keys.
3. Oracle 11g uses standard algorithms—AES and SHA-1.

4.3 HOWTO Use Password Policies

Whenever you create a user, a default password profile is assigned to the user. Password profiles define how Oracle will manage password-related issues—for example, how many failed logon attempts Oracle allows before it locks out the account, what kinds of passwords it accepts (in terms of password complexity), and how long you can keep a password before Oracle forces you to change it. It is your job to ensure that you use secure password profiles and that all users have a secure password profile. This is especially true because the default password policy does not have enough settings to protect you from password-based attacks. To create a password profile:

```
SQL> create profile sec_prof limit
  2 failed_login_attempts 5
  3 password_life_time 60
  4 password_reuse_time 60;
Profile created.
```

What you've defined is a profile by which an account will be locked if there are five continuous failed login attempts, that the password lifetime is 60 days and that the same password cannot be reused within a period of 60 days. You can change the attributes of a password profile, for example:

```
SQL> alter profile sec_prof limit password_reuse_max 10;
Profile altered.
```

When creating a user you assign the password profile to enforce the password protection behavior:

```
SQL> create user myappusr identified by <pwd>;
User created.
SQL> grant create session to myappusr;
Grant succeeded.
SQL> alter user myappusr profile sec_prof;
User altered.
```

The full list of attributes that you can set for a password profile is shown below. In Oracle 11g each parameter that is omitted has a default value also shown:

- FAILED_LOGIN_ATTEMPTS: # of failed logins before an account is locked. The default in 11g is 10.
- PASSWORD_LIFE_TIME: # of days after which a password must be changed before it becomes invalid. The default in 11g is 180.
- PASSWORD_REUSE_TIME: # of days within which the same password cannot be reused (e.g., change to another password and then change it right back). This parameter is used in concert with PASSWORD_REUSE_MAX—that is, you must set values for both. PASSWORD_REUSE_MAX is the # of times that you may reuse a password. The effect on the password profile is as follows:
  - If you specify an integer number for both then you cannot reuse a password until the password has been changed PASSWORD_REUSE_MAX times during PASSWORD_REUSE_TIME days.

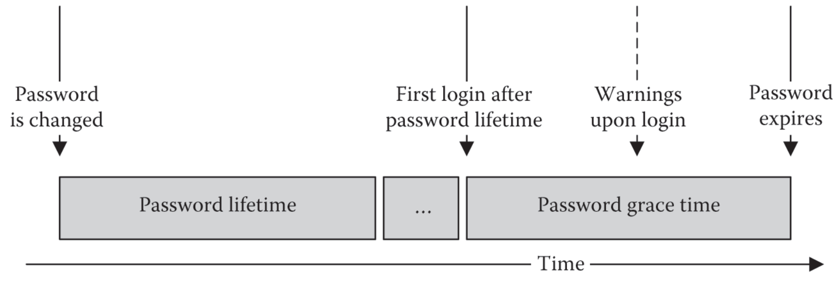


Figure 4.1 Understanding password lifetime and grace time.

  - If one of the parameters is set to UNLIMITED and the other has an integer value then you can never reuse a password.
  - If one of the values is set to DEFAULT then the value is taken from the default password policy.
- If both are set to UNLIMITED then Oracle ignores these values in the password profile. This is the default behavior if you omit these parameters.
- PASSWORD_LOCK_TIME—# of days that must pass after an account is locked out before it is unlocked. The default in 11g is one day.
- PASSWORD_GRACE_TIME—When a password expires this parameter allows for an additional grace period during which a user can logon using the old password. Figure 4.1 depicts what Oracle will do based on the time values you define as the grace period and the lifetime. A warning is issued when you login during the grace period. The default in 11g is seven days.
- PASSWORD_VERIFY_FUNCTION—Allows you to specify a function for validating password strength—you'll learn about this in the next HOWTO.

For all the numerical values you can use UNLIMITED—but you should prefer not to do so (except for PASSWORD_LOCK_TIME). This defeats the purpose of using password profiles which is to protect you from a password-based attack.

To set values for the default profile (the one used for all users unless you specifically alter a user to assign it to another profile), use:

```
SQL> alter profile default limit password_life_time 90;
Profile altered.
```

Finally, one quick mention of a new initialization parameter introduced in Oracle 11g—SEC_MAX_FAILED_LOGIN_ATTEMPTS (with a default value of 10). This parameter (set with an ALTER SYSTEM command) controls the number of authentication attempts that a connection can attempt before the server drops the connection. This is different from FAILED_LOGIN_ATTEMPTS in password profiles. The value in password profiles controls the attempts to logon to a user account. SEC_MAX_FAILED_LOGIN_ATTEMPTS is oblivious to which account the attempt relates to—so if an attack randomly guesses both an account name and a password then after ten attempts the server will drop the connection. Note that no account will be locked—the attacker can initiate a new connection and try ten more. However, dropping and restarting a new connection takes time—so this slows down the attack considerably and allows you to react to it.

Two Things to Remember about Password Policies

1. A password profile is always assigned to a user—even if you don't do so explicitly. The default password profile is used—so set the default profile to ensure all users get the values that comply with your security policy.
2. Don't use UNLIMITED as a value within the password policy. The only exception may be functional accounts used by application connection pools.

4.4 HOWTO Enforce Password Complexity

Passwords should not be easy to guess. Passwords which are easy to guess allow an attacker to gain access to an account. There are many recommendations as far as creating complex passwords; one of the more common guidelines for password strength is

- Passwords should have at least 8 characters and no more than 30 characters. Passwords should never be shorter than 8 characters and 30 characters is the maximum in Oracle.
- Passwords should have at least one uppercase character, one lowercase character, at least one number, and at least one special character.
- Password should not start with a number.
- Passwords should not be a word that can be looked up in a dictionary.
- Passwords should not be constructed using a well-defined scheme.

The last directive in this list requires a little more explanation. Because passwords that adhere to complex policies are not easy to create and are even harder to remember, people have come up with some fairly clever schemes to generate passwords that will conform to these requirements and yet are not hard to remember. As an example, let's look at an example for a scheme that many people use:

- Choose a word that you can easily remember but which is long enough.
- Choose a word that starts with a consonant and uppercase the first letter in the password.
- Start replacing some letters in the word with numbers and special characters in a way that is deterministic and easy to remember. For example, use numbers and special characters that look somewhat like the letters you are replacing—replace A with 4 and S with 5, etc. Don't replace the first letter even if it is a vowel.

Let's look at a few examples:

- Start with the word "malicious" and replace some letters—so your password becomes M4l1c10^5.
- Start with the word supersecret and replace some letters—so your password becomes S^p3r53cr3t.

This sounds like a reasonable scheme, doesn't it? If you think it does, do a quick search on the Web for a program called "John the Ripper" and various variants on the original program—you'll see that people have created password crackers based on such schemes. You should assume that any method that you come up with to generate passwords that will meet these specifications in a way that will be easy to remember have already been devised, and that password guessers based on these method also have been created. Unfortunately, only random passwords are REALLY secure.

As discussed in Section 4.3, you enforce password strengths through the PASSWORD_VERIFY_FUNCTION parameter of password profiles. This parameter gets a value which is a PL/SQL function that is verified every time a user assigned the password profile tries to set a password. If the desired password is not compliant with what this function requires the change will fail.

Oracle provides a sample password verification function in the UTLPWDMG.SQL script located in $ORACLE_HOME/RDBMS/ADMIN. You can use this function as is or use it as a starting point to build a function that is specific to your organization's password requirements. The script file provides two different implementations of a PASSWORD_VERIFY function—one that you can use in Oracle 11g and one that you can use in pre-11g instances.

Two Things to Remember about Password Complexity

1. A good policy is to require passwords to have at least eight characters, at least one uppercase, one lowercase, one number, and one special character.
2. Use the sample UTLPWDMG.SQL script provided as an example for a complexity verification function.

4.5 HOWTO Check for Weak and Default Passwords

Default accounts that come prepackaged with the database and have default and well-known passwords have plagued Oracle (and every other database for that matter). Many of the simplest attacks have been carried out by starting with a logon to a default account with a default password. Oracle 11g has two remedies for this. First, default accounts that come with the database are either set to be locked with an expired password or have to be given a nondefault password when you install the database. This takes care of new installations of the database. But what happens with database that you upgrade from 10g? A new feature of Oracle 11g helps you with this and allows you to continuously monitor if there are any default passwords. Oracle 11g has a new view called DBA_USERS_WITH_DEFPWD. If you select from this view you will get a list of all default accounts that have a default password. As an example, if you have the default SCOTT account set with a password of TIGER and you run:

```
SQL> select * from dba_users_with_defpwd;
```

You will get the result:

```
USERNAME
-----------------
SCOTT
```

Oracle 11g has all the hashes for the default passwords and checks if there are any passwords in USER$ that have the hash values for these default passwords. This makes it very easy for you to make sure that you're not exposed through default passwords. You should periodically (as part of a regular audit) select from this view and make sure that it comes back empty. You can find more resources for checking for default and weak passwords in Section 4.10.

Three Things to Remember about Default Passwords

1. Many experts consider default passwords to be one of the major reasons why attacks occur.
2. In 11g all default accounts come expired and locked by default. This is not the case prior to 11g.
3. Oracle 11g has a view that shows you all accounts with default passwords—make sure to use it periodically.

4.6 HOWTO Set Password Case

Oracle passwords have always been case insensitive. This changed in Oracle 11g. By default, passwords are now case sensitive and therefore stronger. An initialization parameter called SEC_CASE_SENSITIVE_LOGON controls whether or not passwords are case sensitive—the default being TRUE. Set this value to FALSE and restart the database if you need to remain with case insensitive passwords.

If you upgrade a database from Oracle 10g to 11g and you leave the default so that passwords are case sensitive, then the first time users change their passwords the passwords become case sensitive. A user that was defined in 10g will have an uppercase password until they change their password. When they change their password they are free to choose any case for any letter in the password. Furthermore, a new column PASSWORD_VERSIONS in the DBA_USERS view provides information regarding which version of Oracle the password was created in. In the example below the user RON was created in 10g but changed password after the upgrade to 11g, the user JANE was created in 11g and the user RONB was created in 10g and has not changed password since the upgrade. Therefore, RONB still has a fully uppercase password but the passwords for RON and JANE are case sensitive:

```
SQL> select username,password_versions from dba_users;

USERNAME PASSWORD_VERSIONS
------------------------------ -----------------

RON 10G 11G
JANE 11G
RONB 10G
```

Password case sensitivity in Oracle 11g is also supported when you use password files (see Chapter 6). To make a password file support case sensitive passwords use the orapwd utility with the new parameter IGNORECASE = N. If you have a password file that was created in Oracle 10g and you upgrade your database, re-create the password file and regrant the sysdba and sysoper privileges as described in Chapter 6.

In addition to this initialization parameter and the SEC_MAX_FAILED_LOGIN_ATTEMPTS initialization parameter already mentioned in HOWTO 4.3, Oracle 11g introduces a few other initialization parameters that affect account security. These initialization parameters are used as the default password profile:

- FAILED_LOGIN_ATTEMPTS—default is 10.
- PASSWORD_GRACE_TIME—default is 7.
- PASSWORD_LOCK_TIME—default is 1.

- PASSWORD_LIFE_TIME—default is 180.
- PASSWORD_REUSE_MAX—default is UNLIMITED.
- PASSWORD_REUSE_TIME—default is UNLIMITED.

When you start using case-sensitive passwords you need to pay special attention to database links. For credential-based links you need to consider links that go between pre-11g to 11g databases. If you are linking from a pre-11g database the passwords you use for the link user must be all uppercase—because the pre-11g database will uppercase the password when it tries to connect. For links that go from 11g to 11g and for links that go from 11g to pre-11g there is no issue—the first because both sides are case sensitive and the second because the password will be converted to uppercase by the authenticating pre-11g database.

Two Things to Remember about Password Case

1. Passwords in 11g are case sensitive by default.
2. Links defined from pre-11g databases to 11g databases must use an account with an all-uppercase password.

4.7 HOWTO Use Impossible Passwords

You've already seen how password hashes are computed in both Oracle 11g and pre-11g databases. In 11g the password hash is generated by using SHA-1 and in pre-11g a complex scheme that includes encryption. Recall that the last step of this proprietary scheme involves taking the last 8 bytes which form the hash—and converting it into 16 printable hex characters. Saving hashes as printable hex is very natural because a byte has 256 possible values which are represented by two hex digits and using printable hex means that the values can be read by humans.

One implication that follows from schemes that use printable hex characters is that password hashes cannot include certain characters—at least not if they were generated by the algorithm. As an example, a password hash can never include an "S" or a "T" or a "P"—or for that matter any letter apart from A, B, C, D, E, or F. You can use this to your benefit in generating what is called an impossible password. You can do this because when you create a user you can either provide a password or you can directly provide a password hash.

Let's look at an example. The standard way you create a user is to provide a password. In this case Oracle computes the password hash and stores it in USER$:

```
SQL> create user usr1 identified by password;
User created.
SQL> select name,password from sys.user$ where name like 'USR%';

NAME                             PASSWORD
-------------------------------- --------------------------------

USR1                             46A3F68B0AAC78BC

1 row selected.
```

Instead, Oracle allows you to create a user and give a password hash using IDENTIFIED BY VALUES instead of IDENTIFIED in the CREATE USER command. In this case the value that you provide is saved within the USER$ table as is:

```
SQL> create user usr2 identified by values 'password';
User created.
SQL> select name, password from sys.user$ where name like 'USR%';

NAME                                      PASSWORD
---------------------------------  ------------------------------

USR1                                      46A3F68B0AAC78BC
USR2                                      password

2 rows selected.
```

At this point USR2 will never be able to logon to Oracle. Why? Because there is no password on earth that the user can enter that, after applying the SHA-1 or the proprietary algorithm will produce a hash with a value of "password"—there are letters in this value that are not hex.

Why might you want to use impossible passwords? Why would you not just lock an account or drop the user? Usually you would—locking an account is the normal facility that you should normally use. However, there is one subtle difference in terms of the information Oracle provides when someone tries to logon using that user. Lock the USR1 account and try to logon using both USR1 and USR2 to see what happens:

```
SQL> alter user usr1 password expire account lock;
User altered.
SQL> connect usr1/password
ERROR:
ORA-28000: the account is locked
Warning: You are no longer connected to ORACLE.
SQL> connect usr2/password
ERROR:
ORA-01017: invalid username/password; logon denied
```

The subtle difference is the error message produced when the logon is unsuccessful and the information you provide to a potential attacker. When you lock out the account the message produced tells the attacker exactly what is going on. This helps the attacker—they know what the status is and can direct their efforts appropriately. In the second case, when using impossible passwords, you provide absolutely no information—the attacker cannot know if the account exists, if the password is wrong, etc. Less information is better—so use it if you have very special requirements for certain accounts. Incidentally, the ANONYMOUS user in a default installation has an impossible password:

```
SQL> select username,password from dba_users where username='ANONYMOUS';

USERNAME                            PASSWORD
------------------------------------ ---------------------------------

ANONYMOUS                           anonymous

1 row selected.
```

Two Things to Remember about Impossible Passwords

1. Password hashes can only include hex characters.
2. You can set a password hash directly. If you set the hash to have any letter above F no one will ever be able to logon to that account because no password will map to the hash. The error message will not allow an attacker to learn anything about this account.

4.8 HOWTO Limit System Resources Used by Users

In HOWTO 4.1 you saw that when you create a user you can set their quota on the tablespace. From a security point of view, you do this to contain any DoS attack that may be launched from this account. There are many more system resource limits that you can set for an account—and from a security perspective they too should be used to contain a possible DoS attack.

Limiting system resources is done through profiles in the same way that you use password profiles (see HOWTO 4.3). For example, to limit the time a user session is allowed to remain connected and the time a session can be idle you create a profile and assign it to a user as follows:

```
SQL> create profile conn_prof limit
  2 connect_time 240
  3 idle_time 30;
Profile created.
SQL> alter user ronb profile conn_prof;
User altered.
```

Profiles can create both password policy-related parameters as well as resource limits. There is no difference as far as Oracle is concerned—both define boundaries for connections made with a certain user account. Therefore, you can mix these parameters into a single profile as follows:

```
SQL> create profile sec_prof limit
  2 connect_time 240
  3 idle_time 30
  4 failed_login_attempts 5
  5 password_life_time 60
  6 password_reuse_time 60;

Profile created.
```

The list of system resources that you can limit using profiles is shown below. There are three types of resources—connection resources, session resources, and call resources. For connection resources (CONNECT_TIME and IDLE_TIME) if your session exceeds these times the database rolls back the current transaction and ends your session. What you will see is an error when you issue the next command. If you violate a session resource limit Oracle will abort the operation, roll back the current statement and return an error. You can commit or roll back the current transaction but then you need to exit the session. If you violate a call limit Oracle aborts the operation, rolls back the current statement, and returns an error. Each of these values can be set to UNLIMITED or to DEFAULT in which case the value of the default profile is used.

1. CONNECT_TIME—The maximal time in minutes for a session.
2. IDLE_TIME—The maximal time in minutes that a session is allowed to be inactive. Both the CONNECT_TIME and the IDLE_TIME are not continuously checked because this is

a waste of resources. A check is performed every few minutes so the termination of a session can be up to five minutes later than the value you define in the profile.

3. SESSIONS_PER_USER—The maximal number of concurrent sessions allowed per user who receives this profile.
4. CPU_PER_SESSION—The CPU time limit in hundredths of a second that is allowed per session.
5. CPU_PER_CALL—The CPU time in hundredths of a second that is allowed per call.
6. LOGICAL_READS_PER_CALL—The number of data blocks read from memory or disk that is allowed for processing a single SQL statement.
7. LOGICAL_READS_PER_SESSION—The number of data blocks read from memory or disk that is allowed per session.
8. COMPOSITE_LIMIT—The total resource cost in service units that is allowed per session. The COMPOSITE_LIMIT is a weighted sum of CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION, and PRIVATE_SGA and gives you a normalized measure of how many resources the session is consuming.
9. PRIVATE_SGA—The maximal size in the shared pool of the SGA that is allowed per session.

Two Things to Remember about Limiting Resources

1. Use profiles to limit system resources used by user sessions.
2. Limit system resources to combat DoS attacks.

4.9 HOWTO View Information on Users and Profiles

Oracle gives you a number of views for inspecting information that you need to manage users and profiles. The views that you would use most often are

1. DBA_PROFILES—Shows all the profiles and their defined limits. For example:

```
SQL> select * from dba_profiles order by profile;

PROFILE                     RESOURCE_NAME                     RESOURCE    LIMIT
--------------------------  --------------------------------  ----------  ----------

APP_USER2                   COMPOSITE_LIMIT                   KERNEL      DEFAULT
…
APP_USER2                   FAILED_LOGIN_ATTEMPTS             PASSWORD    5
APP_USER2                   PASSWORD_LIFE_TIME                PASSWORD    30
APP_USER2                   PASSWORD_REUSE_TIME               PASSWORD    60
APP_USER2                   PASSWORD_REUSE_MAX                PASSWORD    5
APP_USER2                   PASSWORD_VERIFY_FUNCTION          PASSWORD    DEFAULT
APP_USER2                   PASSWORD_LOCK_TIME                PASSWORD    10
APP_USER2                   PASSWORD_GRACE_TIME               PASSWORD    10
…
CONN_PROF                   IDLE_TIME                         KERNEL      30
CONN_PROF                   CONNECT_TIME                      KERNEL      240
CONN_PROF                   PRIVATE_SGA                       KERNEL      DEFAULT
CONN_PROF                   FAILED_LOGIN_ATTEMPTS             PASSWORD    DEFAULT
…
DEFAULT                     COMPOSITE_LIMIT                   KERNEL      UNLIMITED
DEFAULT                     SESSIONS_PER_USER                 KERNEL      UNLIMITED
…
```

2. DBA_TS_QUOTAS (and USER_TS_QUOTAS for a specific user)—Shows all tablespace quotas currently defined. For example:

```
SQL> select * from dba_ts_quotas;

TABLESPACE_NAME   USERNAME   BYTES     MAX_BYTES    BLOCKS MAX_BLOCKS   DRO
----------------- ---------- --------- ------------ -------------------- ---
SYSAUX            DMSYS      262144    209715200    32                   25600 NO
SYSAUX            SYSMAN     51838976  -1           6328                 -1 NO
USERS             TEST1      65536     1048576      8                    128 NO
SYSAUX            OLAPSYS    16318464  -1           1992                 -1 NO
SYSTEM            SCOTT      0         -1           0                    -1 NO
SYSTEM            TEST1      0         1048576      0                    128 NO

6 rows selected.
```

3. DBA_USERS shows all users in the database (ALL_USERS for what the logged on user can see). For example:

```
SQL> select username, profile, account_status from dba_users;

USERNAME              PROFILE                        ACCOUNT_STATUS
--------------------- ------------------------------ ---------------------
SECURITY_ROLES        DEFAULT                        OPEN
JONES                 DEFAULT                        OPEN
MYAPP                 APP_USER2                      OPEN
HR                    DEFAULT                        OPEN
JANE                  DEFAULT                        OPEN
RONB                  CONN_PROF                      EXPIRED
TSMSYS                DEFAULT                        EXPIRED & LOCKED
BI                    DEFAULT                        EXPIRED & LOCKED
PM                    DEFAULT                        EXPIRED & LOCKED
DBSNMP                MONITORING_PROFILE             OPEN
…
```

You can also use this view to show the password hashes:

```
SQL> select username,password from dba_users;

USERNAME              PASSWORD
--------------------- ------------------
SECURITY_ROLES        952EDE259E4D2BB4
JONES                 5215700C08CDCF93
MYAPP                 C050EF9228761727
HR                    4C6D73C3E8B0F0DA
JANE                  4968BF82C7B085C6
RONB                  35E71E94C51B984C
TSMSYS                3DF26A8B17D0F29F
BI                    FA1D2B85B70213F3
PM                    72E382A52E89575A
DBSNMP                A0070F5A6F1A9AF8
…
```

4. RESOURCE_COST—Shows the cost per resource in terms of CPU, read time, connection time, and SGA size per session.
5. USER_PASSWORD_LIMITS—Shows the password parameters assigned to the user. For example:

```
SQL> connect myapp
Enter password: *****
Connected.

SQL> select * from user_password_limits;

RESOURCE_NAME                            LIMIT
---------------------------------------  ---------
FAILED_LOGIN_ATTEMPTS                    5
PASSWORD_LIFE_TIME                       30
PASSWORD_REUSE_TIME                      60
PASSWORD_REUSE_MAX                       5
PASSWORD_VERIFY_FUNCTION                 NULL
PASSWORD_LOCK_TIME                       10
PASSWORD_GRACE_TIME                      10

7 rows selected.
```

6. USER_RESOUCE_LIMITS—Shows the resource limits for the user. For example:

```
SQL> select * from user_resource_limits;

RESOURCE_NAME                           LIMIT
------------------------------------    ----------------
COMPOSITE_LIMIT                         UNLIMITED
SESSIONS_PER_USER                       UNLIMITED
CPU_PER_SESSION                         UNLIMITED
CPU_PER_CALL                            UNLIMITED
LOGICAL_READS_PER_SESSION               UNLIMITED
LOGICAL_READS_PER_CALL                  UNLIMITED
IDLE_TIME                               30
CONNECT_TIME                            240
PRIVATE_SGA                             UNLIMITED

9 rows selected.
```

7. V$STATNAME—Shows user session statistics.

4.10 Additional Resources

There are many resources on the Web for helping you check for default accounts and default passwords, and many resources that can help check for weak passwords. In Oracle 11g many of that is built into the database but for pre-11g versions these resources can come in very handy:

Lists of default accounts:

- List of default accounts in a normal Oracle 11g installation—See Table 3-1 and Table 3-2 in Oracle 11g Release 1 Security Guide (Part Number B28337-03 Chapter 3).
- Pete Finnigan's list of default passwords (includes many options, modules, and applications above and beyond the base database): www.petefinnigan.com/default/default_password_list.htm

Password checkers/guessers:

- www.cqure.net
- www.red-database-security.com/software/checkpwd.html
- Password checker based on Pete Finnigan's list of default accounts/passwords: www.petefinnigan.com/default/default_password_checker.htm

Examples for writing password verifier functions and scripts for checking password complexity:

- Scripts for checking password complexity: UTLPWDMG.SQL in $ORACLE_HOME/RBDMS/ADMIN. There are two functions—one checking 11g databases and one checking pre-11g databases. Among other things the script checks that
  - Passwords contain at least 8 characters and no more than 30.
  - Passwords are not the same as the user name, the user name spelled backwards, or the user name with numeric characters appended.
  - Passwords are not the same as the server name or the server name with numeric characters added.
  - Passwords are not change_on_install.
  - Passwords are not simple words in a dictionary or dictionary words with numeric characters added.
  - Passwords include at least one numeric and one alphabetic character.
- Examples for verifier function:
  - http://orafaq.com/node/58
  - http://aspalliance.com/746
  - http://www.pentest.co.uk/cgi-bin/viewcat.cgi?cat = downloads

Password management tools (help you remember/store your passwords securely):

- Password safe—http://sourceforge.net/projects/passwordsafe
- Keepass—http://keepass.sourceforge.net

Database security

HOWTO Secure and Audit Oracle 10g and 11g

Ron Ben Natan

Foreword by Pete Finnigan

Oracle is the number one database engine in use today. The fact that it is the choice of military organizations and agencies around the world is part of the company's legacy and is evident in the product. Oracle has more security-related functions, products, and tools than almost any other database engine. Unfortunately, the fact that these capabilities exist does not mean that they are used correctly or even used at all. In fact, most users are familiar with less than 20 percent of the security mechanisms within Oracle.

Written by Ron Ben Natan, one of the most respected and knowledgeable database security experts in the world, ***HOWTO Secure and Audit Oracle 10g and 11g*** shows readers how to navigate the options, select the right tools and avoid common pitfalls. The text is structured as *HOWTOs* — addressing each security function in the context of Oracle 11g and Oracle 10g.

Among a long list of *HOWTOs*, readers will learn to —

- Choose configuration settings that make it harder to gain unauthorized access
- Understand when and how to encrypt data-at-rest and data-in-transit and how to implement strong authentication
- Use and manage audit trails, and advanced techniques for auditing
- Assess risks that may exist and determine how to address them
- Make use of advanced tools and options such as Advanced Security Options, Virtual Private Database, Audit Vault, and Database Vault

The text also provides an overview of cryptography, covering encryption and digital signatures and shows readers how Oracle Wallet Manager and orapki can be used to generate and manage certificates and other secrets.

While the book's 17 chapters follow a logical order of implementation, each *HOWTO* can be referenced independently to meet a user's immediate needs. Providing authoritative and succinct instructions highlighted by examples, this ultimate guide to security best practices for Oracle bridges the gap between those who install and configure security features and those who secure and audit them.

CRC Press
Taylor & Francis Group
an **informa** business
www.taylorandfrancisgroup.com

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487
270 Madison Avenue
New York, NY 10016
2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK

AU4127
ISBN: 978-1-4200-8412-2
90000
9 781420 084122

www.auerbach-publications.com