

Unraveling the Sweater

Oracle Database Security (Part 1)

Tim Gorman – Principal, SageLogix, Inc.

Overview

Some of the most experienced database administrators in the world leave their systems open to casual hacking. Hackers aren't only lonely 13 year olds with bad skin – they could be a co-worker just trying to get his/her job done without getting tangled up in the bureaucratic red tape of change management or data security. It could also be a more malicious co-worker who likes to know things about other co-workers, customers or patients.

The purpose of this article is to show how simple it is to break into most Oracle databases. The point is not to alarm you or alert you to how scary and dangerous I am, but rather to provide justification to take some simple steps to close the easiest security loopholes.

The nature of the problem

Whether immediately after installation or over the years since, you may have noticed a good number of accounts that were created in your Oracle database. If you are running a database version that was created before Oracle9i, then you might see something like this:

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

```
USERNAME
```

```
-----
```

```
SYS
```

```
SYSTEM
```

```
DBSNMP
```

```
SCOTT
```

```
OUTLN
```

```
ORDSYS
```

```
ORDPLUGINS
```

```
MDSYS
```

```
CTXSYS
```

```
PERFSTAT
```

```
AURORA$JIS$UTILITY$
```

```
OSE$HTTP$ADMIN
```

```
AURORA$ORB$UNAUTHENTICATED
```

You didn't create them and they have nothing to do with the applications that you are running. They are used by the various products comprising the Oracle database server, such as the *spatial data option*, the *Oracle Text option* and *9iAS*. Even if you don't intend to use these options, it is still someone's

responsibility to make sure that these accounts don't become *back entrances* into your database, your application and your data.

Almost everyone is aware that the SYS account has the default password of CHANGE_ON_INSTALL and SYSTEM has the default password of MANAGER, and that it is prudent to change both immediately. But did you know that a variety of other *default accounts* are installed, many with well-known default passwords.

In Oracle9i, the situation improves somewhat, as the majority of these *default accounts* are created *locked* and *expired*:

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

USERNAME	ACCOUNT_STATUS
-----	-----
SYS	OPEN
SYSTEM	OPEN
DBSNMP	OPEN
SCOTT	OPEN
OUTLN	EXPIRED & LOCKED
WMSYS	EXPIRED & LOCKED
ORDSYS	EXPIRED & LOCKED
ORDPLUGINS	EXPIRED & LOCKED
MDSYS	EXPIRED & LOCKED
CTXSYS	EXPIRED & LOCKED
...and so on...	

Moreover, in Oracle9i the passwords for SYS and SYSTEM accounts can be set during the execution of the CREATE DATABASE command and the *Database Creation Assistant (DBCA)* of the *Oracle Universal Installer (OUI)* contains logic to prompt for non-default passwords.

So, you might think that the problem is fully resolved. Not at all...

Unraveling the sweater – starting the thread...

There is an old saying amongst both burglars as well as police:

Secrets are like a pretty woman's sweater – find a loose thread, start pulling, and don't stop pulling until the sweater is gone.

Well, the actual saying is a lot more profane, but that's the general idea. To be politically correct, this axiom works for all combinations of genders, the owner of the sweater does not have to be attractive, and of course the sweater itself could be made from hemp or other non-animal products.

Most successful burglars (and police investigators) are patient, willing to exploit any flaw in order to learn enough to take the next step, to move closer to their objective. In this case, find a loose thread.

Even though it may not lead anywhere, gently start pulling. You never know, it might lead to something productive!

It only takes one unguarded account to start the complete unraveling of data security. That one account may not have important permissions by itself, but just about every default has basic READ permissions for a lot of important information. Take the example of an account named TEST that was created on an Oracle9i Release 2 (v9.2.0.1) database with only CREATE SESSION privileges:

```
SQL> select object_type, count(*) from dba_objects
      2  group by object_type;
select object_type, count(*) from dba_objects group by
object_type
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

This is all well and good. An account that has only CREATE SESSION privileges should certainly not be permitted to even know the existence of the DBA-level data dictionary views. Now, how about:

```
SQL> select object_type, count(*) from user_objects
      2  group by object_type;

no rows selected
```

OK, this makes sense. Every account is permitted to see what objects it owns. In this case, for an account with just the CREATE SESSION privilege, there couldn't possibly be any objects because the permissions to create them have not been granted.

So far, so good. Now, what if we query:

```
SQL> select object_type, count(*) from all_objects
      2  group by object_type;

OBJECT_TYPE          COUNT(*)
-----
CONSUMER GROUP              2
EVALUATION CONTEXT         1
FUNCTION                   89
INDEXTYPE                   8
JAVA CLASS                 9654
JAVA RESOURCE              184
LIBRARY                    14
OPERATOR                   27
PACKAGE                   267
PROCEDURE                  14
SEQUENCE                    2
SYNONYM                   11589
```

```

TABLE          46
TYPE           536
VIEW          1102

```

15 rows selected.

Gee, that's a lot of information we can look at because, as you know, the ALL_OBJECTS data dictionary view only displays objects for which you have been granted at least one permission, such as SELECT. This means that the innocent little account called TEST, which was granted the very barest minimum of permissions, is capable of viewing information contained in *thousands* of objects.

So, one little unprotected account with the barest minimum of permissions really is a security problem also. What about these unprotected default accounts, like DBSNMP, which is used by the Oracle Enterprise Manager product. This account has the default password of DBSNMP, even in Oracle9i. It is granted the CONNECT role as well as the SELECT ANY DICTIONARY permission. Wow! You can look at a lot of information when you can view absolutely any data dictionary view. Sure hope that some developer didn't embed any usernames or passwords into any stored procedures, huh?

So what, you ask? Big deal! So you can view information; it doesn't mean that you can hurt anything, does it?

OK. Let's look at one example, the ALL_USERS data dictionary view:

```
SQL> select * from all_users;
```

```

USERNAME                                USER_ID  CREATED
-----
SYS                                       0  12-MAY-02
SYSTEM                                   5  12-MAY-02
OUTLN                                    11 12-MAY-02
DBSNMP                                   19 12-MAY-02
WMSYS                                    21 12-MAY-02
ORDSYS                                    30 12-MAY-02
ORDPLUGINS                               31 12-MAY-02
MDSYS                                    32 12-MAY-02
CTXSYS                                    33 12-MAY-02
XDB                                       35 12-MAY-02
ANONYMOUS                                36 12-MAY-02
WKSYS                                    39 12-MAY-02
WKPROXY                                   40 12-MAY-02
ODM                                       42 12-MAY-02
ODM_MTR                                  43 12-MAY-02
OLAPSYS                                   44 12-MAY-02
TEST                                      62 07-NOV-02

```

...(and so on for dozens or hundreds of lines)...

With this view, we can get the names of all of the user-accounts in the database. Every single account in the database can view this information! Now, if we got far enough into this database by guessing passwords on just one default account, don't you think that it's a good bet we can do it for another account? Knowing the name of the account may help, as guessing passwords is really not all that difficult.

Guessing passwords

It's not really that hard. The key to good password management is to make it arbitrary but memorable. But many people don't bother. If you know a little about a person, then you have more options for guessing passwords, such as the names of sports teams, musical artists, pets, children.

People can be quite predictable. Everyone knows that you're supposed to have both numbers as well as alphabetic letters in your password, so everyone makes the same substitutions:

- The letter "o" becomes the number "0"
- The letters "i" and "l" become the number "1" or "7"
- The letter "e" becomes the number "3"
- The letter "g" becomes the number "6"
- Follow the original word with a number

It is extremely common to use the name of the account as the password. Many people choose the word "oracle" as their Oracle account password. Since Oracle is not case sensitive for passwords, there is no need to worry about variations in upper or lower case. It doesn't take much to make this more complex, but hackers (whether internal or external) rely on people being *lazy* or *unaware*.

The *Oracle Security Handbook*, by Marlene Theriault and Aaron Newman, published by Oracle Press in 2001 (ISBN #0-07-213325-2), has a listing of the most common *default user accounts* from Oracle7 through Oracle9i. It is not hard to figure out the list of these default user accounts on your own, as anyone can download the latest version of Oracle from <http://otn.oracle.com>, install it and then examine the contents of the ALL_USERS view immediately after installation.

I highly recommend this book for a complete treatment of Oracle database security issues.

The sweater unravels...

So, depending on the version of the RDBMS you are using and depending on what products you have installed (whether they be Oracle products or third party products), there are dozens of these *default user accounts* that might be tried. If you script the process of trying different passwords on each of them, it should be rather easy to learn a great deal.

For example:

- If you are able to connect to the SYS account using the password CHANGE_ON_INSTALL, then you are dealing with a very lazy and foolish database administrator.
- If you are able to connect to the SYSTEM account using the password MANAGER, then the same conclusion applies as above.

- If you are able to connect to any account using the name of the account as the password, then the database administrator (like almost everyone else!) has not bothered to utilize any of the features of password management that have been part of the RDBMS since version 8.0 was introduced in 1997.
- If accounts like MDSYS return a message of *ORA-01017: invalid username/password; logon denied* instead of *ORA-28000: the account is locked*, then this might suggest you are attempting to connect to an RDBMS of version 8.1.x or below, since Oracle9i and above locks this account by default.
- If you are able to repeatedly try various passwords on an account name and receive only the *ORA-01017: invalid username/password; logon denied* message, then there is no way to know whether you are getting the account name right or wrong. However, it means that you can keep trying different password possibilities almost indefinitely.
- If the database administrator has enabled account locking after a number of FAILED_LOGIN_ATTEMPTS, then you will eventually start to receive the *ORA-28000: the account is locked* message, which indicates that you have at least gotten the account name correct. However, the fact that the account is now locked may require manual intervention from the database administrator, so he/she may not be aware of failed attempts to log in.

I have posted a UNIX shell script named *oraprobe.sh* on my website at <http://www.EvDBT.com/tools.htm>, located toward the bottom of that page, which uses these techniques to automate the probing of a database's security. The script expects as a command-line parameter a TNS *connect-string* for connecting to the database instance. It will then iterate through a list of *default user accounts* and attempt several different well-known passwords against each.

If it is able to guess a password correctly, then the script will log in to the database and query the ALL_USERS database view to get a list of all account names. The script will then attempt to log in to the database using the list of valid account names by attempting to use the account name for the password.

So, you see in this brief, simple shell-script a bit of the concept of finding one loophole in security and then exploiting it for further advantage.

Patently unraveling the sweater.

Summary

Neither this paper nor the *oraprobe.sh* script is intended to encourage malicious behavior. It is intended to point out some extremely simple security loopholes that can exist in all versions of Oracle.

If you are familiar with the *strong passwording* features introduced in Oracle8 v8.0 in 1997, then you know that many of the bad habits can be reduced or eliminated once and for all. These features allow you to create *user profiles* (i.e., CREATE PROFILE, ALTER PROFILE) that can then be assigned to users using CREATE USER or ALTER USER. Characteristics that can be assigned include:

- FAILED_LOGIN_ATTEMPTS = *number of attempts allowed before locking account*
- PASSWORD_LIFE_TIME = *number of days password can be valid*
- PASSWORD_REUSE_TIME = *number of days password cannot be reused*

- PASSWORD_REUSE_MAX = *number of times a password can be reused*
- PASSWORD_LOCK_TIME = *number of days account remains locked*
- PASSWORD_GRACE_TIME = *number of days before password expires*
- PASSWORD_VERIFICATION_FUNCTION = *PL/SQL function to verify mandatory password complexity (example provided online at "\$ORACLE_HOME/rdbms/admin/utlpwdmg.sql" on all platforms)*

Of course, you have to be cautious in using many of these *strong passwording* features, as it is important to understand the impact that they might have on the applications that are utilizing Oracle.

Also, there is a balancing act to be aware of, as enthusiastic adoption of *strong passwording* can severely inconvenience and alienate end users and developers, causing them to create larger password-security breaches by writing their passwords down. So, forcing users to remember more than two or three passwords to get their daily jobs done will result in more serious security problems.

But some of these mechanisms are quick and painless. Setting the passwords of unused (or seldom-used) accounts to unspeakably long and nonsense strings is just good housekeeping. Locking the account using ALTER USER ... ACCOUNT LOCK is useful, too.

In addition to the judicious use of *strong passwording* features, this paper will hopefully also alert you to the usefulness of Oracle's database auditing features. The AUDIT SESSION command will, by default, create an audit trail entry for every *unsuccessful* as well as every *successful* connection attempt. Periodic analysis of the DBA_AUDIT_SESSION data dictionary view will reveal any repeated unsuccessful connection attempts.

If you have a UNIX or Linux machine anywhere on your network, use the *oraprobe.sh* shell-script to test your basic security. If you have ideas about possible passwords that people in your company might use, then add them to the list of possible passwords used inside the shell script. The results from the script may surprise you and lead the way into a discussion of better securing your valuable data.

About part 2 of this paper

This paper was primarily about encouraging the use of some of the features of *strong passwording* and *database audit trails*. The second half of this article will be about simple steps in protecting the TNS Listener service both from unauthorized usage as well as providing valuable information about your server topology to the casual observer.

Biography

Tim Gorman works for SageLogix, a small consulting and managed-services company focusing on Oracle database servers and related I/O and backup subsystems (<http://www.SageLogix.com>). He is a database administrator with a focus on Oracle internals, performance tuning, security, troubleshooting, and high availability. He has been a DBA since a dark and stormy night in 1993, an Oracle developer since 1990, and a "C" programmer since 1983. He is co-author, with Gary Dodge, of *Essential Oracle8i Data Warehousing* (Sept 2000) and *Oracle8 Data Warehousing* (March 1998), both published by John Wiley and Sons.