

Database Security: Principle of Least Privilege

Mike Henderson, October 2007

About the author

Mike Henderson has a justifiable reputation as one of the top Oracle consulting DBAs in Ireland. His time with Curach Consulting has supplemented a distinguished 18 year career in IT including several years with Oracle Corporation. Mike's roles have encompassed development, database administration and consultancy, but the ability to deliver optimised database performance through expert and innovative design is his hallmark. Mike is an Oracle Certified Professional and currently has identified database security, high availability and Oracle Text as areas for promotion.

Introduction

'The price of peace is eternal vigilance'. Whilst the original intent of the passage was not database security the quote holds true for every DBA. Security has always been a concern for the DBA but the growth of compliance requirements over recent years has put database security high on the agenda.

This series of Insight pieces is intended to provide a guide to improving the security of Oracle databases. We will be taking small steps up from the standard install and taking each concept a little further than is regularly applied. The hints here should be applicable to any database edition from 9i up and usually earlier versions too. The next Insight piece will look at passwords and user profile management.

Extending the principles of least privilege

DBAs and developers should be familiar with one of the foundations of database security, the principle of least privilege. For those unfamiliar with it, the principle is that users should only be granted the minimum amount of privilege in order to conduct their assigned tasks.

The practical application of this principle is usually expressed as by the creation of a series of database roles e.g. a viewer role, an 'accounts role' that can perform certain functions but not other, an HR role with a different series of grants & privileges etc. A role is effectively a container for object grants and database privileges.

In an Oracle Forms environment or similar, roles usually comprise grants on tables. When a user is granted the role they have direct access rights on a table, select grant allows them to view the whole of any row in that table, updates can be applied to any column of the table, deletes can be to any row. The privileges are general and unconstrained - this cannot be 'least privilege'!

In an api driven environment direct access to tables is not allowed and is replaced with a series of execute rights on packages. There will be an insert procedure, another for update, maybe a few with different select properties. The api constrains the access and may enforce extra business logic, limiting privilege and enhancing security. This model is generally considered more secure than the direct grant approach.

So far so good, but what we haven't addressed are dimensions such as time, location and application.

Curach Consulting - The Academy, 42 Pearse Street, Dublin 2.
Tel: + 353 1 2459000 Fax: + 353 1 2459001 Web: www.curach.com

ENABLING HIGHER
BUSINESS PERFORMANCE



In most environments, roles are enabled by default when the login succeeds. A user gains full rights anytime they manage to connect to the database. If we apply the principle of least privilege then this method is probably not appropriate - the roles that the user was given were intended for use within the HR package, not for when they started up a SQL Plus session. Neither method above differentiates. Oracle has addressed this problem in a number of ways, with varying degrees of usefulness.

The first mechanism is the non default role - a role which has been granted but has to be explicitly asked for before it becomes active.

```
dba_sql> CREATE ROLE BIG_ROLE;
dba_sql> GRANT UPDATE ON EMP TO big_ROLE;
dba_sql> GRANT BIG_ROLE TO mike;
dba_sql> ALTER USER MIKE DEFAULT ROLE ALL EXCEPT big_role;
```

This mechanism works fine embedded in scripts for sql*Plus but can be difficult to embed in some application tools e.g. Reporting tools. This mechanism is only effective as long as the user does not realise there are non default roles, once the secret is out it ceases to be a barrier to inappropriate activity.

```
mike_sql> SET ROLE big_ROLE;
```

Password protected roles are a logical next step but their implementation is dogged by all the standard problems inherent in password management such as secrecy and the effort required in changing a compromised password. One of the biggest security concerns is the 'Insider Threat', inappropriate staff activity. Securing passwords in such circumstances can be a large overhead.

```
dba_sql> create role big_role2 identified by bigpass;
dba_sql> grant big_role2 to mike;
dba_sql> alter user mike default role all except big_role2;
```

```
Mike_sql>set role bigrole2 identified by bigpass;
```

Oracle 9i introduced the secure application role. This is basically the ability to grant privilege based on the outcome of a pl/sql package. The package is custom built and is free to use any authentic factor available. This could be the time of day, the ip address of the request, the state of a certain parameter.

```
dba_sql> create role bigrole3 identified using
utils.manage_bigrole;
dba_sql> grant bigrole3 to mike;
dba_sql> alter user mike default role low_privs;
utils_sql> create or replace procedure manage_bigrole authid
current user
as
begin
```

Curach Consulting - The Academy, 42 Pearse Street, Dublin 2.
Tel: + 353 1 2459000 Fax: + 353 1 2459001 Web: www.curach.com

ENABLING HIGHER
BUSINESS PERFORMANCE



```
if (to_char(sysdate,'dd') < 25 then
  dbms_session.set_role(bigrole3);
end if;
end;
/
```

```
utils_sql> grant execute on manage_bigrole to mike;
mike_sql> exec utils.manage_bigrole
```

So here we have a simple procedure that will only grant the bigrole3 role if the date is less than the 25th of the month. It could easily check the time of day or the ip address of the sender, for example. The point is that you are now in control not only of what a role can do but also when and from where it can be invoked.

Loopholes

Loophole 1: Owner=User

The best way to beat any role-based security structure is not to use it. A table owner's access to their own table cannot be limited or hindered by roles as they implicitly have full control. This creates a very large hole in the application of the principle. In extreme cases applications use a pooled or shared login connecting as the application owner, relying totally on any middle tier privilege mechanism. This shortcuts any layered security strategy and should be avoided at all costs.

A simple solution would be to make the rule that table owners do not connect to the database. There should be no need as DML can be performed by any authorised user and DDL can be performed by any DBA. Revoke the create session privilege and audit connects by the user to make sure that it does not change.

Loophole 2: Any Any

There are a group of system privileges that can be used to override object privileges - the ANYs, e.g. update ANY table. Much favoured by developers in a hurry, they are directly contrary to the principle of least privilege and should be avoided.

The information contained herein is provided by Curach Consulting and is intended to provide general information only and is not intended to constitute legal, consulting or other professional advice or services.

Curach Consulting - The Academy, 42 Pearse Street, Dublin 2.
Tel: + 353 1 2459000 Fax: + 353 1 2459001 Web: www.curach.com

ENABLING HIGHER
BUSINESS PERFORMANCE

