UKOUG DBMS SIG, March 17th 2009

Using Oracle VPD In The Real World

By

Pete Finnigan

Updated Friday, 24th February 2009
Why Am I Qualified To Speak

- PeteFinnigan.com Limited
- Founded February 2003
- CEO Pete Finnigan
- Clients UK, States, Europe
- Specialists in researching and securing Oracle databases providing consultancy and training
- [http://www.petefinnigan.com](http://www.petefinnigan.com)
- Author of Oracle security step-by-step book
- Published many papers, regular speaker (UK, USA, Slovenia, Norway, Iceland and more)
- Member of the Oak Table Network
Agenda

• What is VPD? is it used? info?
• Differences in various Oracle versions
• Securing VPD – often not considered
• Attacking VPD
• Problems – performance – design
• Conclusions

“Whilst VPD is a security solution, security solutions must also be secured themselves and unfortunately they also increase the attack surface”
What Is VPD – Many Names.

• Called Virtual Private Database (VPD)
• Called Row Level Security (RLS)
  – Hence DBMS_RLS controls it
• Called Fine Grained Access Control (FGAC)
• VPD includes:
  – Fine Grained Access Control
  – Application Contexts
  – Global Application Contexts

“Used by Oracle themselves in Label Security (the Trusted Oracle replacement) and also Database Vault”
Is VPD Used In Anger?

• In my experience not much – why?
  – I have worked with a few clients to implement VPD
  – It is free with EE; not a cost option that may put people off like OLS

• Oracle are increasingly using it
  – In XDB ACL’s
  – In E-Business Suite
  – As part of Database Vault and Audit Vault
Where To Find Information

- Oracles on-line documentation
- Effective Oracle database 10g security by design - ISBN-13: 978-0072231304
VPD Through The Versions

• Row Level Security added in 8.1.5 release
• 9i adds multiple policies per table and policy groups controlled by application driving context
• 9i adds global contexts for connection pooling
• 10g adds column level policies, column masking, policy types (5) added for performance to allow caching, contexts updated to allow values to be passed to parallel slaves.
• 11g provides integration for Enterprise manager for Row Level Security Policies.
Securing VPD

- Leaking predicates
- Leaking policies
- “R”ole “B”ased “AC”cess (RBAC) on VPD structure / configuration
- Bypassing VPD by means of exception
- SQL Injection issues
- Direct data access

“Remember: An important concept in using security features is to ensure that the security feature itself is also secure”
Finding the Predicate

- There are a number of possibilities to find predicates and details
  - Event 10730
  - Event 10060
  - V$vpd_policy – no one has access by default
- Library cache dump? – if static data present can also be leaked
- SGA can be dumped for binds, SQL, optimizer and more
- Common denominator – ALTER SESSION / SYSTEM / trace (many options - http://www.petefinnigan.com/ramblings/how_to_set_trace.htm)
Create A Simple Policy

- See code
  [http://www.petefinnigan.com/vpd2.sql](http://www.petefinnigan.com/vpd2.sql)

- Create a user PXF,
  - Grant some privileges,
  - Create a table (copy of scott.emp)
  - Create a predicate function to block “deptno != 10”
  - Create a policy on pxf.emp
  - Number of rows restricted by 3
Example

```
PRIVILEGE TO CHECK [SELECT ANY TABLE]: ALTER SESSION
OUTPUT METHOD Screen/File [S]: S

Privilege => ALTER SESSION has been granted to =>

Role => DBA (ADM = YES) which is granted to =>
  User => SYS (ADM = YES)
  User => SYMAN (ADM = NO)
  User => SYSTEM (ADM = YES)
  User => TESTUSER (ADM = NO)
User => SYS (ADM = NO)
User => IX (ADM = NO)
User => SH (ADM = NO)
Role => RECOVERY_CATALOG_OWNER (ADM = NO) which is granted to =>
  User => SYS (ADM = YES)
User => BI (ADM = NO)
User => CTXSYS (ADM = NO)
Role => OLAP_USER (ADM = NO) which is granted to =>
  User => SYS (ADM = YES)
  User => SCOTT (ADM = NO)
User => HR (ADM = NO)
User => DMSYS (ADM = NO)
User => XDB (ADM = NO)
```
SQL> alter session set sql_trace=true;
Session altered.
SQL> alter session set events '10730 trace name context forever';
Session altered.
SQL> select * from pxf.emp;

EMPNO   ENAME      JOB               MGR   HIREDATE         SAL
--------- ----- -------------- ------ -------------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
---------- ---------- ---------- --------- ---------- -------
----------- 7369 SMITH      CLERK           7902 17-DEC-80        800 20  

As a normal user – SCOTT - I am able to determine the rules VPD imposes on me
SQL> alter session set events '10730 trace name context off';
Session altered.
SQL> alter session set sql_trace=false;
Session altered.
SQL>
### Example (3)

<table>
<thead>
<tr>
<th>SQL Trace File</th>
<th>Timestamp</th>
<th>SQL Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ora10gr2_ora_7784.trc</td>
<td>18/03/2009</td>
<td><code>alter session set sql_trace=true</code></td>
</tr>
</tbody>
</table>
Leaking Policy Details

- To secure VPD all of the configuration must be secured including:
  - `%_POLICY_GROUPS`
  - `%_POLICY_CONTEXTS`
  - `%_POLICIES`

- Access to predicate functions must also be protected;
  - Definitions – OBJ$, SOURCE$, PROCEDURE$, ARGUMENT$, the functions
Example

NAME OF OBJECT TO CHECK [USER_OBJECTS]: ALL_POLICIES
OWNER OF THE OBJECT TO CHECK [USER]: SYS
OUTPUT METHOD Screen/File [S]: S
FILE NAME FOR OUTPUT [priv.lst]:
OUTPUT DIRECTORY [DIRECTORY or file (/tmp)]:
EXCLUDE CERTAIN USERS [N]:
USER TO SKIP [TEST%]:

Checking object => SYS.ALL_POLICIES
==================================================================================================

Object type is => VIEW (TAB)
Privilege => SELECT is granted to =>
Role => PUBLIC (ADM = NO)

Madness by default anyone can see what policies exist that affect them
Example(2)

SQL> select object_owner, object_name, policy_name,
2       pf_owner, pf_owner, function
3   from all_policies;

<table>
<thead>
<tr>
<th>OBJECT_OWNER</th>
<th>OBJECT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOTT</td>
<td>EMP</td>
</tr>
<tr>
<td>PXFPOL</td>
<td>SYS</td>
</tr>
<tr>
<td>SYS</td>
<td>HACK</td>
</tr>
<tr>
<td>PXF</td>
<td>EMP</td>
</tr>
<tr>
<td>PXFTEST</td>
<td>PXF</td>
</tr>
<tr>
<td>PXF</td>
<td>PREDICATE</td>
</tr>
</tbody>
</table>

2 rows selected

As SCOTT I could find out the predicate, I can also find out the policies that affect me.

Bad design!
RBAC on VPD structure

- RBAC must be applied on
  - Packages – DBMS_RLS, DBMS_SESSION
  - Policies – see previous slide
  - Policy functions, structure, source code
  - Contexts, application and global
  - Supporting data – static look up data – Any data used in a policy/predicate
  - System privileges used
  - Grants on access to any of the above
- Don’t just rely on VPD to protect data
Bypassing VPD

• VPD configuration should be designed normally to work with users (end users / identities)
  – i.e. access to groups of data is based on actual people, this is reflected in the VPD
• This is often done in total or part using application contexts – These are tied to the session
• BUT, they must use static data, session data, application data (i.e. FND_PROFILES) to ascertain who is who
• Whilst the context is reasonably secure often the data used could be changed/bypassed/spoofed
• All of the identity must be considered and hardened
Exempt Access Policy

who_has_priv: Release 1.0.3.0.0 - Production on Wed Jan 16 16:26:56 2008
Copyright (c) 2004 PeteFinnigan.com Limited. All rights reserved.

PRIVILEGE TO CHECK [SELECT ANY TABLE]: EXEMPT ACCESS POLICY
OUTPUT METHOD Screen/File [S]: S
FILE NAME FOR OUTPUT [priv.lst]:
OUTPUT DIRECTORY [DIRECTORY or file (/tmp)]:
EXCLUDE CERTAIN USERS [N]:
USER TO SKIP [TEST%]:

Privilege => EXEMPT ACCESS POLICY has been granted to =>
========================================================================
User => X (ADM = NO)

PL/SQL procedure successfully completed.

For updates please visit http://www.petefinnigan.com/tools.htm

SQL> http://www.petefinnigan.com/who_has_priv.sql
• SQL Injection could be used in a number of ways to exploit VPD:
  – Litchfield shows how to inject a call to DBMS_RLS.DROP_POLICY via XDB.XDB_PITRIG_PKG.PITRIG_DROP – see http://www.databasecurity.com/dbsec/ohh-defeating-vpd.pdf
  – Many exploits from sites such as http://milw0rm.com can be used in the same way
  – Packages that expose VPD – see next slide
  – Applications that VPD could have components exploited – i.e. if the predicate is “constructed” using concatenation it could be exploited.
### Ways To Access Policies

```sql
SQL> select owner, name, type
    2   from dba_dependencies
    3   where referenced_name='DBMS_RLS';

<table>
<thead>
<tr>
<th>OWNER</th>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC</td>
<td>DBMS_RLS</td>
<td>SYNONYM</td>
</tr>
<tr>
<td>SYS</td>
<td>DBMS_RLS</td>
<td>PACKAGE BODY</td>
</tr>
<tr>
<td>SYS</td>
<td>LTUTIL</td>
<td>PACKAGE BODY</td>
</tr>
<tr>
<td>SYS</td>
<td>LTADM</td>
<td>PACKAGE BODY</td>
</tr>
<tr>
<td>XDB</td>
<td>DBMS_XDBZ0</td>
<td>PACKAGE BODY</td>
</tr>
<tr>
<td>XDB</td>
<td>DBMS_XDBZ0</td>
<td>PACKAGE BODY</td>
</tr>
</tbody>
</table>
```

```sql
SQL> select grantee, table_name from dba_tab_privs
    2   where table_name in ('LTUTIL','LTADM','DBMS_XDBZ0');

<table>
<thead>
<tr>
<th>GRANTEE</th>
<th>TABLE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMSYS</td>
<td>LTADM</td>
</tr>
<tr>
<td>WMSYS</td>
<td>LTUTIL</td>
</tr>
<tr>
<td>IMP_FULL_DATABASE</td>
<td>LTADM</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>DBMS_XDBZ0</td>
</tr>
</tbody>
</table>
```
Access The Data Directly

- Strings on data files
- With C or Java from the database
- Hex editors – Unix or Windows
- Block dumps – recent forensics papers cover
- Tools like bbed, CBAT, DUL like tools such as Ora*Dude and more
- Backups
- Exports
- Reports and lists of data from privileged users
- More?

Again do not consider VPD as a “be all” and and “end all” – work out where the data is and how it “flows”
Example (1)

```sql
SQL> select distinct dbms_rowid.rowid_block_number(rowid) blk,
        2     dbms_rowid.rowid_relative_fno(rowid) fno
        3  from pxf.emp;

<table>
<thead>
<tr>
<th>BLK</th>
<th>FNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>420</td>
<td>4</td>
</tr>
</tbody>
</table>

1 row selected.

SQL> select file_name from dba_data_files
    2  where file_id=4;

FILE_NAME
---------
C:\ORACLE\ORADATA\ORA10GR2\USERS01.DBF

1 row selected.
```
Example (2)

SQL> alter system dump datafile 4 block 420;
System altered.
SQL> connect sys/change_on_install as sysdba
Connected.
SQL> select * from pxf.emp where deptno=10;

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>JOB</th>
<th>MGR</th>
<th>HIREDATE</th>
<th>SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7782</td>
<td>CLARK</td>
<td>MANAGER</td>
<td>7839</td>
<td>09-JUN-81</td>
<td>2450</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7839</td>
<td>KING</td>
<td>PRESIDENT</td>
<td></td>
<td>17-NOV-81</td>
<td>5000</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7934</td>
<td>MILLER</td>
<td>CLERK</td>
<td>7782</td>
<td>23-JAN-82</td>
<td>1300</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Example (3)

```
<table>
<thead>
<tr>
<th>Repeat 463 times</th>
</tr>
</thead>
<tbody>
<tr>
<td>8229DC0 00000000 0B002C00 2350C203 4C494D06 [..........P#.MIL]</td>
</tr>
<tr>
<td>8229DD0 0552454C 52454C43 4EC2034B B6770753 [LER.CLERK..NS.w.]</td>
</tr>
<tr>
<td>8229DE0 01D17101 0EC2020D 0B102100 0308042C [..........]</td>
</tr>
<tr>
<td>8229DF0 040350C2 45243F46 414E4107 5453594C [P., FORD, ANALYST]</td>
</tr>
<tr>
<td>8229E00 434CC020 0CB57707 0101D100 FF1FC202 [..LC.w..........]</td>
</tr>
<tr>
<td>8229E10 2C15C102 2C202080 21A60550 0553454D [........P.JAMES]</td>
</tr>
<tr>
<td>8229E20 52454D43 4DC2034B B5770763 0101D10C [CLERK..Mc.w.....]</td>
</tr>
<tr>
<td>8229E30 0A2C3010 C102F333 00002C1F 4D4FC203 [........3..OM]</td>
</tr>
<tr>
<td>8229E40 41444105 4305534D 4B52454C 594EC203 [ADAMS.CLERK..NY]</td>
</tr>
<tr>
<td>8229E50 05BB7707 0101D117 FF0ECC20 2C15C102 [w.............]</td>
</tr>
<tr>
<td>8229E60 C2034080 54062D4F 454E5255 41530852 [D_.TURNER-SA]</td>
</tr>
<tr>
<td>8229E70 4D53454C C2034E41 7707634D 010809B5 [LESMAN..Mc.w.....]</td>
</tr>
<tr>
<td>8229E80 C3034010 02060116 0202C1C0 4C203406 [........0]</td>
</tr>
<tr>
<td>8229E90 494B0428 5034474E 42534542 544E5544 [..KING.PRESIDENT]</td>
</tr>
<tr>
<td>8229EA0 B57707FF 0101D100 33C20201 0B102100 [w........3..]</td>
</tr>
<tr>
<td>8229EB0 0308002C 0554EC02 544F4353 4E410754 [.....NY.SCOTT.AN]</td>
</tr>
<tr>
<td>8229EC0 53549C41 4CC20354 B6770743 0101D104 [ALYST..LC.w.....]</td>
</tr>
<tr>
<td>8229ED0 1FC20201 15C102F3 0308002C 05534EC2 [..........NS]</td>
</tr>
<tr>
<td>8229EE0 52414C43 414D074B 4547441E 4FC20352 [CLARK.MANAGER..Q]</td>
</tr>
<tr>
<td>8229EF0 B5770728 0101D106 31C20301 C102F333 [aw.........3..]</td>
</tr>
<tr>
<td>8229F00 0308002C 0416C2C2 44524E57 4C415308 [L..WARD.PAL]</td>
</tr>
<tr>
<td>8229F10 47414D41 C2035245 7707284F 0101D105 [ANAGER..O(w.....]</td>
</tr>
<tr>
<td>8229F20 C2034010 02FF331D 002C1FC1 4C203408 [..........3..M]</td>
</tr>
<tr>
<td>8229F30 414D0637 4E495452 4C415308 414D5345 [7.MARTIN.SALESMAN]</td>
</tr>
<tr>
<td>8229F40 4D2034E1 B5770763 0101D109 0DC20301 [N..Mc.w.....]</td>
</tr>
<tr>
<td>8229F50 FC20233 2C1FC102 C2030800 4A05534C [0NES.MANAGER..Q]</td>
</tr>
<tr>
<td>8229F60 53454E4F 4B414D07 52454741 284FC203 [w...........L]</td>
</tr>
<tr>
<td>8229F70 0B577077 0101D102 41EC203 15C102F3 [w.........3..]</td>
</tr>
<tr>
<td>8229F80 0308002C 0416C2C2 44524E57 4C415308 [3.MANAG.E.PAL]</td>
</tr>
<tr>
<td>8229F90 414D5345 4DC2034B B5770763 0101D102 [K.D.ALEN.SALESMAN]</td>
</tr>
</tbody>
</table>
```

18/03/2009

Copyright (c) 2009 PeteFinnigan.com Limited

25
• Certification and Support for Third party products - [http://blogs.oracle.com/schan/newsItems/departments/extendingApps/2006/05/18#a200](http://blogs.oracle.com/schan/newsItems/departments/extendingApps/2006/05/18#a200)
• Adding VPD can break existing applications and other modules
• E-Business Suite screens have been seen to break because VPD is enabled
• There is often a fear with VPD implementers that they are not supported if VPD breaks something
• You can get into a complex support / certification saga
• If Oracle can reproduce – even if you let support have your code or an example with the same problem Oracle can help look at the issue
Layered Approach

- VPD must be part of a layered approach to securing data in an Oracle database
- RBAC on
  - Data
  - Security measures and policies
- Encryption for critical data
- Hardening must be done
- VPD as part of an overall solution
- Network security
- Audit trails
- More…
• VPD is often perceived as being bad due to perceived optimizer changes – aim to not excessively change the optimizer
• Often runs faster when VPD is enabled – less rows returned!
• Don’t use excessive code in predicates i.e. select from dual or worse big tables
• Use indexes on the predicate columns
• Use static data if at all possible
• Use static policies if possible
• Keep the policy functions as simple as possible – good design is king!
Another lesson learned was to pass back `sys_context('...', '...')` rather than resolve the `sys_context` in the policy function.

5 types of caching can be used:
- **Static** – execute once, store predicate in SGA
- **Shared_static** – cache predicate across multiple objects using same policy
- **Context_sensitive** – use for connection pooling, server executes policy function on statement execution if a context change detected
- **Shared_context_sensitive** – as above; shared across multiple objects; same policy
- **Dynamic** – no caching executed every time
One of the key lessons I have learned with VPD is to design carefully first. Include:

- Business rules first (who/what/when)
- Identify the data to be protected
- Simplicity is the key – keep the rules / policies very simple (as simple as possible)
- Work out the identities, the rules for all access, the default state,
- Then design the contexts, predicates
- Test – create boundary tests as well
Multiple Policy Issues

- An example from the trenches
- A single table is needed as part of every predicate
- A lot of other tables access this table as part of the predicate generation
- A lot of policies created, identities designed, contexts created
- Problem: The single table cannot be protected with VPD as it breaks all other policies
- VPD needs, hardening, RBAC etc as well as a “complete” solution
Conclusions

- Looked at “what is VPD”
- What can it do
- How VPD can be bypassed and why
- How the data could still be accessed outside of VPD
- How to design VPD implementations
- How to protect VPD implementations
Any Questions?
Contact - Pete Finnigan

PeteFinnigan.com Limited
9 Beech Grove, Acomb
York, YO26 5LD

Phone: +44 (0) 1904 791188
Mobile: +44 (0) 7742 114223
Email: pete@petefinnigan.com