

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>
- 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**
- RLS chapter - <http://www.devshed.com/c/a/Oracle/RowLevel-Security-with-Virtual-Private-Database/>
- Does VPD, FGA or audit really cause performance issues - <http://www.insight.co.uk/files/presentations/Does%20VPD,%20FGA%20or%20Audit%20Cause%20Performance%20Issues.pdf>
- Oracle Row Level Security - <http://www.securityfocus.com/infocus/1743>
- Row Level Security - <http://www.dbazine.com/oracle/or-articles/jlewis15>

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)

“In Oracle security we must think in layers: the object, the meta-data, the access rights, different paths to the data or to the “right”....”

Create A Simple Policy

- See code
<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

who_has_priv: Release 1.0.3.0.0 - Production on Wed Jan 16 19:13:16 2008
Copyright (c) 2004 PeteFinnigan.com Limited. All rights reserved.

```
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 => SYSMAN (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)
```

Example (2)

```
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)

```
TextPad - [C:\app\Admin\diag\vdms\ora11gpe\ora11gpe\trace\ora11gpe_ora_2936.trc]
File Edit Search View Tools Macros Configure Window Help

*** 2009-03-17 09:35:43.734
*** SESSION ID:(134.69) 2009-03-17 09:35:43.734
*** CLIENT ID:( ) 2009-03-17 09:35:43.734
*** SERVICE NAME:(SYS$USERS) 2009-03-17 09:35:43.734
*** MODULE NAME:(SQL*Plus) 2009-03-17 09:35:43.734
*** ACTION NAME:( ) 2009-03-17 09:35:43.734

=====
PARSING IN CURSOR #3 len=32 dep=0 uid=81 oct=42 lid=81 tim=1962102740 hv=0 ad='c2dfc04' sqlid='00000000000000'
alter session set sql_trace=true
END OF STMT
EXEC #3:c=0,e=37,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,tim=1962102735
=====
PARSING IN CURSOR #4 len=59 dep=0 uid=81 oct=42 lid=81 tim=1962166757 hv=2603787413 ad='0' sqlid='8scgalqdm594p'
alter session set events '10730 trace name context forever'
END OF STMT
PARSE #4:c=0,e=31,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,tim=1962166753
EXEC #4:c=0,e=145,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,tim=1962167069
=====
PARSING IN CURSOR #4 len=39 dep=1 uid=483 oct=47 lid=483 tim=1962168080 hv=1410854723 ad='2813f5a0' sqlid='0f4xhatalgvu3'
begin :con := PREDICATE(:sn, :on); end;
END OF STMT
PARSE #4:c=0,e=32,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=1962168075
EXEC #4:c=0,e=91,p=0,cr=0,cu=0,mis=0,r=1,dep=1,og=1,tim=1962168387
=====
Logon user      : SCOTT
Table/View     : PXF.EMP
Policy name    : PXFTEST
Policy function: PXF.PREDICATE
RLS view      :
SELECT "EMPNO", "ENAME", "JOB", "MGR", "HIREDATE", "SAL", "COMM", "DEPTNO" FROM "PXF"."EMP" "EMP" WHERE (deptno != '10')
=====
PARSING IN CURSOR #4 len=450 dep=1 uid=81 oct=3 lid=81 tim=1962169557 hv=2160055283 ad='2813f130' sqlid='4xsj2xq0bznzm'
SELECT /* OPT_DYN_SAMP */ /*+ ALL_ROWS IGNORE WHERE_CLAUSE NO_PARALLEL(SAMPLESUB) opt_param('parallel_execution_enabled', 'false') NO_PARALLEL_INDEX(SAMPLES)
END OF STMT
PARSE #4:c=0,e=307,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=1,tim=1962169551
EXEC #4:c=0,e=856,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=1,tim=1962170590
FETCH #4:c=0,e=112,p=0,cr=3,cu=0,mis=0,r=1,dep=1,og=1,tim=1962170777
STAT #4 id=1 cnt=1 pid=0 pos=1 obj=0 op='SORT AGGREGATE (cr=3 pr=0 pw=0 time=0 us)'
STAT #4 id=2 cnt=15 pid=1 pos=1 obj=79720 op='TABLE ACCESS FULL EMP (cr=3 pr=0 pw=0 time=2 us cost=3 size=4251 card=327)'
=====
PARSING IN CURSOR #3 len=21 dep=0 uid=81 oct=3 lid=81 tim=1962171373 hv=3160365295 ad='28142c0c' sqlid='1u2fc12y5yq7g'
select * from pxf.emp
END OF STMT
PARSE #3:c=0,e=3725,p=0,cr=4,cu=0,mis=1,r=0,dep=0,og=1,tim=1962171368
=====
PARSING IN CURSOR #4 len=39 dep=1 uid=483 oct=47 lid=483 tim=1962171612 hv=1410854723 ad='2813f5a0' sqlid='0f4xhatalgvu3'
begin :con := PREDICATE(:sn, :on); end;
END OF STMT
```

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

```
who_can_access: Release 1.0.3.0.0 - Production on Wed Jan 16
                19:30:16 2008
```

```
Copyright (c) 2004 PeteFinnigan.com Limited. All rights reserved.
```

```
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
```

```
=====
===
```

Madness by default anyone can see what policies exist that affect them

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

Example(2)

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

OBJECT_OWNER	OBJECT_NAME
-----	-----
POLICY_NAME	PF_OWNER
-----	-----
PF_OWNER	FUNCTION
-----	-----
SCOTT	EMP
PXFPOL	SYS
SYS	HACK
PXF	EMP
PXFTEST	PXF
PXF	PREDICATE

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

- 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.databasesecurity.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> select owner,name,type
      2  from dba_dependencies
      3  where referenced_name='DBMS_RLS';
```

OWNER	NAME	TYPE
PUBLIC	DBMS_RLS	SYNONYM
SYS	DBMS_RLS	PACKAGE BODY
SYS	LTUTIL	PACKAGE BODY
SYS	LTADM	PACKAGE BODY
XDB	DBMS_XDBZ0	PACKAGE BODY
XDB	DBMS_XDBZ0	PACKAGE BODY

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

GRANTEE	TABLE_NAME
WMSYS	LTADM
WMSYS	LTUTIL
IMP_FULL_DATABASE	LTADM
PUBLIC	DBMS_XDBZ0

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 “end all” – work out where the data is and how it “flows”

Example (1)

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

BLK	FNO
420	4

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;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL
7782	CLARK	MANAGER	7839	09-JUN-81	2450
10					
7839	KING	PRESIDENT		17-NOV-81	5000
10					
7934	MILLER	CLERK	7782	23-JAN-82	1300
10					

Example (3)

```
TextPad - [C:\oracle\admin\ora10gr2\udump\ora10gr2_ora_10228.trc]
File Edit Search View Tools Macros Configure Window Help
Repeat 464 times
8229DC0 00000000 08002C00 2350C203 4C494D06 [.....P#.MIL]
8229DD0 0552454C 52454C43 4EC2034B B6770753 [LER.CLERK..NS.w.]
8229DE0 01011701 0EC20201 0BC102FF 0308002C [.....]
8229DF0 040350C2 44524F46 414E4107 5453594C [..P..FORD.ANALYST]
8229E00 434CC203 0CB57707 01010103 FF1FC202 [..LC.w.....]
8229E10 2C15C102 C2020800 414A0550 0553454D [.....P.JAMES.]
8229E20 52454C43 4DC2034B B5770763 0101030C [CLERK..Mc.w.....]
8229E30 0AC20301 C102FF33 08002C1F 4D4FC203 [....3.....OM]
8229E40 41444105 4305534D 4B52454C 594EC203 [..ADAMS.CLERK..NY]
8229E50 05BB7707 01010117 FF0CC202 2C15C102 [..w.....]
8229E60 C2030800 54062D4F 454E5255 41530852 [.....O-.TURNER.SA]
8229E70 4D53454C C2034E41 7707634D 010809B5 [LESMAN..Mc.w....]
8229E80 C2020101 02800110 002C1FC1 4FC20308 [.....O]
8229E90 494B0428 5009474E 49534552 544E4544 [(.KING.PRESIDENT]
8229EA0 B57707FF 0101110B 33C20201 0BC102FF [..w.....3....]
8229EB0 0308002C 05594EC2 544F4353 4E410754 [.....NY.SCOTT.AN]
8229EC0 53594C41 4CC20354 BB770743 01011304 [ALYST..LC.w.....]
8229ED0 1FC20201 15C102FF 0308002C 05534EC2 [.....NS.]
8229EE0 52414C43 414D074B 4547414E 4FC20352 [CLARK.MANAGER..O]
8229EF0 B5770728 01010906 19C20301 C102FF33 [(.w.....3....]
8229F00 08002C0B 634DC203 414C4205 4D07454B [.....Mc.BLAKE.M]
8229F10 47414E41 C2035245 7707284F 010105B5 [ANAGER..O(.w....]
8229F20 C2030101 02FF331D 002C1FC1 4DC20308 [.....3.....M]
8229F30 414D0637 4E495452 4C415308 414D5345 [7.MARTIN.SALESMA]
8229F40 4DC2034E B5770763 01011C09 0DC20301 [N..Mc.w.....]
8229F50 0FC20233 2C1FC102 C2030800 4A05434C [3.....LC.J]
8229F60 53454E4F 4E414D07 52454741 284FC203 [ONES.MANAGER..O(]
8229F70 04B57707 01010102 4C1EC203 15C102FF [..w.....L....]
8229F80 0308002C 04164CC2 44524157 4C415308 [.....L..WARD.SAL]
8229F90 414D5345 4DC2034E B5770763 01011602 [ESMAN..Mc.w.....]
8229FA0 0DC20301 06C20233 2C1FC102 C2030800 [.....3.....]
8229FB0 4105644B 4E454C4C 4C415308 414D5345 [Kd.ALLEN.SALESMA]
8229FC0 4B52454C 4E454C4C 4C415308 414D5345 [.....]
8229FD0 4E454C4C 4C415308 414D5345 414D5345 [.....]
8229FE0 4E454C4C 4C415308 414D5345 414D5345 [.....]
8229FF0 4E454C4C 4C415308 414D5345 414D5345 [.....]
55 45 Read Ovr Block Sync Rec Caps
```

Screens Can Break

- Certification and Support for Third party products - <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...

Performance

- 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!

Cached Policies and sys_context

- 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

Design It First

- 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


```
create or replace function log_start(fv_path
return utl_file.file_type is
  lv_fptr utl_file.file_type:=null;
  lv_module varchar2(100):='log_start';
begin
  Oracle Security Expertise
  dbms_output.disable;
```

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