You Don’t Have Database Vault

So, What Can You Do Instead?
Pete Finnigan – Background, Who Am I?

• Oracle Security specialist and researcher
• CEO and founder of PeteFinnigan.com Limited in February 2003
• Writer of the longest running Oracle security blog
• Oracle ACE for security
• Member of the OakTable
• Speaker at various conferences
  – UKOUG, PSOUG, BlackHat, more..
• Published many times, see
  – http://www.petefinnigan.com for links
• Influenced industry standards
  – And governments
Agenda

• Part 1
  • What is Database Vault?
  • What does Database Vault do?
  • Components of Database Vault?

• Part 2
  • What can we do to simulate the features of Database Vault without Database Vault
  • What is possible for free?
What Is Database Vault?

- Declarative security framework to allow fine grained access to database objects (tables, views, procedures…) grouped into realms
- Literally unlimited context based security rules can be added to control access to any (well almost any) database object or command
- Default use is to protect against SYSTEM ANY privileges
- Because it is “built-in” to the database kernel it is harder to bypass
- Pre-built shipped realms protect risky parameter changes and the data dictionary and more
- Separation of duties are added by default by creation of a security administrator, user account manager and vault owner
- SYS, SYSTEM and DBA are restricted in value
What Is New In 12c In Database Vault

- Pre-installed software (DV and OLS)
- Works with Multitenant
  - DV must be enabled in the root container before a pluggable container
  - Management with common accounts or delegated or local
- Mandatory realms to protect against direct grants and object owner
  - Was possible in 11g but only with very complex rules
- Privilege analysis allows discovery of used or not used rights
- Simple basic hardening is better in 12.1.0.2 and 12.2.0.1 core database
- Shipped policies for products such as E-Business Suite and SAP and Peoplesoft
- Unified audit trail and default audit for DV and OLS
Default Basic Hardening

- When DV is installed Oracle does some basic hardening and securing automatically for you
- This is described here - https://docs.oracle.com/database/121/DVADM/dv_impact.htm#DVADM70123
- If you are in a multitenant database the hardening is applied to the root container and all pluggable containers are affected
  - If you do not want DV in a PDB and do not agree with these changes you must put them back manually; there are also issues with RAC nodes where manual hardening is needed in some cases on other nodes
- The changes include
  - Parameters changed
  - DBA, IMP_FULL_DATABASE, EXECUTE_CATALOG_ROLE, SCHEDULER_ADMIN,
  - UTL_FILE EXECUTE revoked from PUBLIC
  - ALTER / CREATE / DROP on USER / PROFILE restricted
- SYS and SYSTEM cannot change passwords anymore
The Main DV Components

- Factors
  - Individual elements to use in rules (e.g. IP Address)
- Rules
  - True/False questions for the database
- Rule Sets
  - Groups of rules (Also results in True/False – with AND/OR)
- Realms
  - Protect database objects (uses rules, factors)
- Command Rules
  - Protect access to SQL commands (e.g. CONNECT) (uses rules, factors)
- Secure Application Roles (SAR)
  - Protective access to enable a role (uses rules, factors)
Some limits:
- DBMS_CRYPTO, UNLIMITED TABLESPACE, UTL_FILE
- Run time rights, Maintenance, Create
- Duplicate Rights not distinguished

```sql
SQL> select sys_priv, os_user, module, used_role, sys_priv, obj_priv, object_owner, object_name, object_type, path from dba_used_prv
2 /

<table>
<thead>
<tr>
<th>SYS_PRIV</th>
<th>OS_USER</th>
<th>MODULE</th>
<th>USED_ROLE</th>
<th>SYS_PRIV</th>
<th>OBJ_PRIV</th>
<th>OBJECT_OWNER</th>
<th>OBJECT_NAME</th>
<th>OBJECT_TYPE</th>
<th>PATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE SESSION</td>
<td>apache</td>
<td><a href="http://b0e159bof.localdomain">http://b0e159bof.localdomain</a> (TNS V1-V3)</td>
<td>ORACLE</td>
<td>CREATE</td>
<td>SELECT</td>
<td>PUBLIC</td>
<td>CREATE SESSION</td>
<td>TABLE</td>
<td>GRANT_PATH('PUBLIC')</td>
</tr>
<tr>
<td>CREATE SESSION</td>
<td>apache</td>
<td><a href="http://b0e159orablog12">http://b0e159orablog12</a> (TNS V1-V3)</td>
<td>PUBLIC</td>
<td>CREATE</td>
<td>SELECT</td>
<td>PUBLIC</td>
<td>CREATE SESSION</td>
<td>TABLE</td>
<td>GRANT_PATH('PUBLIC')</td>
</tr>
<tr>
<td>CREATE SESSION</td>
<td>apache</td>
<td><a href="http://b0e159orablog12">http://b0e159orablog12</a> (TNS V1-V3)</td>
<td>PUBLIC</td>
<td>CREATE</td>
<td>SELECT</td>
<td>PUBLIC</td>
<td>CREATE SESSION</td>
<td>TABLE</td>
<td>GRANT_PATH('PUBLIC')</td>
</tr>
</tbody>
</table>

5 rows selected.

<table>
<thead>
<tr>
<th>SYS_PRIV</th>
<th>ROLENAME</th>
<th>SYS_PRIV</th>
<th>OBJ_PRIV</th>
<th>OBJECT_OWNER</th>
<th>OBJECT_NAME</th>
<th>OBJECT_TYPE</th>
<th>PATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE ANY CONTEXT</td>
<td>CREATE ANY CONTEXT</td>
<td>EXECUTE</td>
<td>SYD</td>
<td>DBMS_CRYPTO</td>
<td>PACKAGE</td>
<td>GRANT_PATH('PUBLIC')</td>
<td></td>
</tr>
<tr>
<td>CREATE PROCEDURE</td>
<td>CREATE PROCEDURE</td>
<td>WRITE</td>
<td>SYD</td>
<td>ORACLE</td>
<td>DIRECTORY</td>
<td>GRANT_PATH('PUBLIC')</td>
<td></td>
</tr>
<tr>
<td>CREATE VIEW</td>
<td>CREATE VIEW</td>
<td>EXECUTE</td>
<td>SYD</td>
<td>UTL_HTTP</td>
<td>PACKAGE</td>
<td>GRANT_PATH('PUBLIC')</td>
<td></td>
</tr>
<tr>
<td>CREATE VIEW</td>
<td>CREATE VIEW</td>
<td>SELECT</td>
<td>IMPORTER</td>
<td>C3</td>
<td>VIEW</td>
<td>GRANT_PATH('PUBLIC')</td>
<td></td>
</tr>
<tr>
<td>UNLIMITED TABLESPACE</td>
<td>UNLIMITED TABLESPACE</td>
<td>SET CONTAINER</td>
<td>SET CONTAINER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SET CONTAINER</td>
<td>SET CONTAINER</td>
<td>CREATE TYPE</td>
<td>CREATE TYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREATE INDEXTYPE</td>
<td>CREATE INDEXTYPE</td>
<td>CREATE TRIGGER</td>
<td>CREATE TRIGGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREATE TRIGGER</td>
<td>CREATE TRIGGER</td>
<td>CREATE PROCEDURE</td>
<td>CREATE PROCEDURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREATE SEQUENCE</td>
<td>CREATE SEQUENCE</td>
<td>CREATE PROCEDURE</td>
<td>CREATE PROCEDURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREATE CLUSTER</td>
<td>CREATE CLUSTER</td>
<td>CREATE TABLE</td>
<td>CREATE TABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

18 rows selected.
```
My Sample Application Architecture

- Oracle Linux
- Oracle Database
- Applications (Front Facing Website, back office customer processing)

Server 1 – Linux 5.9

Oracle database schemas
Orablog Schema

Server 2 – Linux 5.9

Apache / PHP / OCI8
Orablog: Web Based CMS Application

Server 3 – Linux 5.9

Apache / PHP / OCI8
BOF: Back Office web based Application

End Users

Power Users

DBA, DEV and Application Support Users

© Copyright 2018 PeteFinnigan.com Limited. All rights reserved. Tel 0044 (0) 7759277220, http://www.petefinnigan.com and info@petefinnigan.com
Data Domains – BAD!!

- All data, front and back office are in the same schema; ORABLOG
- All functionality for front and back office are in the same schema
- The web application and back office users connect to the schema
Hacking My Sample Database / Applications

- Three levels of Hacking
  - As a website un-authenticated user
  - As a database user with just CREATE SESSION
  - As a DBA
Hacking The Sample Database With Realm

---

```
x%')|))a)/*/union/**select/**/33,1,timestamp('27-OCT-13'),timestamp('27-OCT-13'),''CardNumber-''||first_name'||''-''||last_name'||''-''||orablog_crypto.decrypt(pan),'x',0,null,'publish','open','open',null,'name',null,null,timestamp('27-OCT-13'),timestamp('27-OCT-13'),null,0,null,0,null,0,6/**/from/**/orablog.credit_card--
```

As an un-authenticated web user

Same Hack, same results as with no DV !!
As a low power database user

Connect to the database as a user with just CREATE SESSION and exploit a vulnerable package (CUSTA) owned by ORABLOG and read card details

```
pause
exec orablog.custa('x' union select username from all users--');
exec orablog.custa('x' union select orablog.bof_kkrc.dr(cc34) from orablog.bof_pay_details--');
prompt press any key to continue....
```

PL/SQL procedure successfully completed.

name:=[3742345698766678]
nname:=[4049877198543457]

PL/SQL procedure successfully completed.

press any key to continue....

• Low privileged database user can see data in the BOF application
Connect as a DBA with the DBA role and simply select credit card details – no hacking needed as we use SYSTEM ANY

```
SQL> select * from orablog.bof_pay_details;
select * from orablog.bof_pay_details
  *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
SQL> prompt decrypt the cards
decrypt the cards
SQL> select name_on_card,orablog.bof_kkrc.dr(cc34) pan
  2  from orablog.bof_pay_details;
from orablog.bof_pay_details
  *
ERROR at line 2:
ORA-01031: insufficient privileges
```

- DV has some effect BUT only for SYSTEM ANY
Add A Mandatory Realm To ORABLOG Instead

Hmmm, the apps are now broken; we need to add ORABLOG to the realm but it defeats the object; if we hack the database again; same result

```sql
SQL> exec dbms_macadm.delete_realm('BOF Realm');
PL/SQL procedure successfully completed.
SQL> SQL> -- create the BOF realm
SQL> begin
  2   dbms_macadm.create_realm(
  3       realm_name => 'BOF Realm',
  4       description => 'Protect BOF objects',
  5       enabled => dbms_macadm.g_yes,
  6       audit_options => dbms_macadm.g_realm_audit_fail,
  7       realm_type => 3);
  8 end;
  9 /
PL/SQL procedure successfully completed.
SQL> SQL> -- add the objects to the realm
SQL> begin
  2   dbms_macadm.add_object_to_realm(
  3       realm_name => 'BOF Realm',
  4       object_owner => 'ORABLOG',
  5       object_type => 'X',
  6       object_name => 'X');</n  7 end;
  8 /
PL/SQL procedure successfully completed.
SQL> select name, realm_type from dvsys.dv_realm;
```
Hacking My Sample Database / Applications

- Three levels of Hacking
  - As a website un-authenticated user
  - As a database user with just CREATE SESSION
  - As a DBA
- Different Attack Types:

<table>
<thead>
<tr>
<th></th>
<th>Web user</th>
<th>CREATE SESSION</th>
<th>DBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>No DV</td>
<td>Can Read CC</td>
<td>Can Read CC</td>
<td>Can Read CC</td>
</tr>
<tr>
<td>DV OOTB</td>
<td>Can Read CC</td>
<td>Can Read CC</td>
<td>Can Read CC</td>
</tr>
<tr>
<td>DV Realm on CREDIT_CARD and Crypto</td>
<td>Can Read CC</td>
<td>Can Read CC</td>
<td>BLOCKED</td>
</tr>
<tr>
<td>DV Mandatory Realm on CREDIT/Card and Crypto</td>
<td>BROKEN</td>
<td>BLOCKED</td>
<td>BLOCKED</td>
</tr>
</tbody>
</table>
DV Command Rule - Results

The rules are not perfect as we have implemented properly only for Orablog and not BOF but BOF has no client tools installed

The client_program_name is not set from the server so we have used instead Module – but it would be better to use the hash

Implementing factors, rules, rule sets and command rules or rule sets for realms is a large task when a lot of controls are needed
Duct Tape?

• Is Database Vault really duct tape?
  • Most sites have/use bad data security designs; excessive rights, lack of data access controls
  • DV could be seen as duct tape to prevent these bad designs (threats) becoming risks
• At its core, DV is solving issues that could be solved differently
  • Design least rights – revoke privileges – do not use System ANY
  • SoD can be done with careful design of users and other simple protections
• Partly issues are caused also by process; “way of working”
What If: No Database Vault Available?

• If we do not have DV or it is not possible (i.e. SE/SE1/SE2) what can we do?
  • Replicate the technical features of DV?
  • Remove as much of the “problem” as possible that is solved by Database Vault?
• Start with a good security design
  • Aim for least rights
  • Aim for lock down
  • Aim for proper data access controls
  • Add context based security without DV
• Do not use defaults
• Consider application design changes
  • Code and data access levels
What Do We Need To Do To Replicate DV?

- There are a lot of features in DV that we could use: Declarative API’s, factors, realms, rules, SARs, Command rules and within these protect objects, commands, SoD, parameters and much much more…
- If we focus on three simple tasks to consider for replication:
  - SET ROLE, DBMS_SESSION.SET_ROLE to be able to create a SAR
  - ALTER SYSTEM to be able to detect a parameter change
  - System ANY to detect use of SELECT ANY TABLE (for instance)
- There is no way (supported) to “Trap” SET ROLE, ALTER SYSTEM or SELECT ANY TABLE
- ALTER SYSTEM is DDL But it is not trapped by a DDL trigger
- There is no simple way to detect SELECT and act upon it in real time
- Some actions can be detected such as CREATE, ALTER, DROP and most DDL
- There are many gaps in available techniques in a core database to replicate Database Vault
We Need a Select Trigger

- There are limited options to capture a SELECT or SELECT ANY
  - FGA handler (needs EE so not for SE/SE1/SE2/XE)
  - Materialised View (needs views on everything)
  - VPD policy function (Again EE)
  - Trigger on AUD$
- Even more limited options for some actions such as SET ROLE or ALTER SYSTEM
  - So we could use a trigger on AUD$
- Note 72460.1 – This note is no longer available but talked about moving AUD$ tablespace and user and adding triggers BUT
  - This note states it is not supported to do this
  - BUT, DV install moves AUD$ to SYSTEM up to 11.2 but not 12c
Blocking A Select Statement

```sql
-- create a trigger on system.aud$ for select on credit_card
create or replace trigger sys.stk_aud_sel
after insert on system.aud$
for each row
begin
    if (:new.obj$name='CREDIT_CARD' and :new.|action#=3) then
        raise application_error(-20077,'You are not allowed to read this table');
    end if;
-- exception
-- when others then
-- null;
end;
/
```

SQL> connect orablog/orablog@//192.168.56.85:1521/bfora.localdomain
Connected.
SQL> select * from credit_card;
select * from credit_card
*  
ERROR at line 1:
ORA-02002: error while writing to audit trail
ORA-00604: error occurred at recursive SQL level 1
ORA-20077: You are not allowed to read this table
ORA-06512: at "SYS.STK_AUD_SEL", line 3
ORA-04088: error during execution of trigger
A Secure Application Role in SE

```sql
create trigger sys.stk_aud_sar
after insert on system.aud$
for each row
begin
  if (:new.action EQ 55) then
    -- check for a SAR
    declare
      lv_proc varchar2(200);
      lv_res number;
      sar_failed exception;
    pragma exception_init(sar_failed,-20078);
    begin
      select role_proc into lv_proc
      from system.stk_sar_tab
      where role_name=:new.obj$role;
      -- if lv_proc was found then execute it
      execute immediate 'begin :val:=||lv_proc||';end' using out lv_res;
    except
      when sar_failed then
    raise_application_error(-20079,'SAR Check Failed --'||sqlerrm);
    end if;
    exception
      when others then
    null;
  end if;
end;
```

```sql
create function system.stk_rdef_sar return number as
begin
  select sys_context('USERENV','IP_ADDRESS') into lv_ip from dual;
  if(lv_ip='192.168.56.2') then
    return 1;
  else
    return 0;
  end if;
end;
```
But What Are We Really Trying to Achieve?

- Are we really trying to replicate DV in its technical functionality?
- Or are we really trying to replicate the results of applying DV?
- Or even do better?
- **YES, We want to replicate the results not the technical design**
- We can achieve this with:
  - Careful security design
  - Some code
  - Privilege management especially around SYS, SYSTEM, DBA…
- We can do context based security without DV
- What is the risk trying to simulate DV?
  - Should be low provided we have a good base design anyway
Base: Good Security Design

- DV needs a good security base to start with
  - So does non DV, whether DV is eventually used or not
- This should include:
  - Data domains
  - Separation of function from data
  - Separate critical data from non
  - Separate critical function from non
- Least User rights
- Data access controls
- Hardening and patching
Hack The Locked Down System

• This is the same database and applications setup as was used in the DV examples
• Except:
  • The database, OS and Network are locked down
  • The data design has changed to secure the data from the connected user
  • The application code is still vulnerable
• Lets try the same hacks as before

Demo!
• The web and normal user fail
• The DBA still works
• Fix? Revoke ANY from the Orablog DBA role
Why Do We (Perceive We) Need System ANY

- **Needed for development/deployment of code?**
  - Solutions used often is SYSTEM ANY for deployment as it is simple
  - There is no grant select on orablog.tables.* so system ANY is a good replacement BUT gives access to all data (except SYS)
- What other solutions exist:
  - Log on as the schema to deploy code
  - Use SYSTEM ANY but via a schema/protected PL/SQL API that you create – complex and hard to maintain
  - Direct grants on the schema objects but issues arise
    - How to create new objects in the same schema
    - Maintainability of rights
  - **Proxy to the schema**
Cont’d

- Two types of rights via SYSTEM ANY
  - Object change/create (CREATE ANY PROCEDURE)
  - Data access and data change (SELECT ANY TABLE)
- Should the release person be able to change data?
  - No, BUT maybe release require data changes
  - Reading data – probably not
- In general
  - Interactive users should not have SYSTEM ANY
  - Schemas should not have SYSTEM ANY
  - A DBA can work around not having SYSTEM ANY
  - Core accounts such as SYS, SYSTEM, DBA – don’t use
Context: View Based Security

- We can create VIEW BASED security to limit access to read data
  - A PL/SQL function allows tests to be made to check whether access is allowed or not
  - We could also check in this PL/SQL whether the privilege used is SELECT ANY by checking the users actual rights
  - This can block some ANY privileges
- **BUT** system ANY for select can access the base table.

**Solution:**
- Revoke system ANY except for sys
- Block SYSDBA access – The first versions of DV did this
Context: DML Based Security

- This is a simple demo to show that we can apply the same “Realm” type ideas to block DML.
- This cannot be overridden as this is added to the base table and this is not view based.
- Again we could check for System ANY in the PL/SQL code by looking at the callers rights.
- We can also make a mandatory realm – in part at least.

Demo – context_dml.sql
Context: Code Based Security

- We also do not need DV to add context based security to PL/SQL code
- DV has the advantage that it is declarative and does not need code to be hand written
- BUT we can still add context based checks to our code where needed
- This example shows that we can limit a function that gets an encryption key from storage to only be called from its protective API
- In a real system we would also obfuscate and protect the PL/SQL
- When you implement DV a lot of work is still needed anyway
Separation of Duties (SoD)

- Separation of Duties does not need DV to enforce it
- Even with DV real people and database accounts need to be designed and a SoD matrix created to ensure separation exists for all interactive users
- Identify and make decisions on separation
  - Account Manager, Audit Trail Admin, Security Admin, Audit Viewer
- All of these can be implemented with design, least privilege
- Custom DBA role should be created
- SYSTEM should be locked, SYS should be blocked out as SYSDBA
- Reduce, remove SYSTEM ANY
- Use technical solutions to enforce security – DDL, ALTER… type system triggers
- Accountability and audit are needed
Context: Blocking Parameter Changes

- Limit ALTER SYSTEM
- Audit use of ALTER SYSTEM
- Limit even from the DBA (should have custom role anyway and limited rights – NOT DBA, SYSDBA)
- Release SYS when needed but audit use of account
- Triggers on database start and stop to detect that a parameter has changed whilst database is up? – put it back?
- We could also protect spfile with chattr to make the file immutable but only on Linux
Demo – connect to the database as ORABLOG from the web server

Command Rule: Block SQL*Plus - Webserver

```sql
program,
    os_user
  ) values (lv_username,lv_ip_address,lv_program,lv_os_user);
commit;

if(lv_ip_address not in('192.168.56.91','192.168.56.89','192.168.56.1','192.168.56.85','192.168.56.90')) then
  -- the IP address is not allowed
  insert into stk_login_error (login_date,error_line) values (sysdate,1);
  commit;
  RAISE_APPLICATION_ERROR(-20070,'NOT AUTHORISED FROM THIS HOST');
else
  -- test for web server and not apache and not httpd
  if((lv_ip_address in('192.168.56.89')) and
     (upper(lv_program)<>'HTTPD@OELS90ORABLOG.LOCALDOMAIN (TNS V1-V3)') and
     (upper(lv_os_user)<>'APACHE')) then
    -- web server and not httpd and not apache OS user
    insert into stk_login_error (login_date,error_line) values (sysdate,2);
    commit;
    RAISE_APPLICATION_ERROR(-20071,'NOT AUTHORISED WITH THESE DETAILS');
  else
    -- we must be on the admin PC or the actual database server
    insert into stk_login_error (login_date,error_line) values (sysdate,3);
    commit;
    end if;
end if;
-- record that we got here
insert into stk_login_error (login_date,error_line) values (sysdate,4);
commit;
exception
  when others then
    insert into stk_login_error (login_date,error_line) values (sysdate,5);
    commit;
    RAISE_APPLICATION_ERROR(-20073,sqlerrm);
  --
end login_db;
```

- We can perfectly replicate the protection we had in DV with a logon trigger
- We can also use valid node checking but this is not granular
- In this example the httpd still works but SQL*Plus from the webserver is blocked
Privilege Analysis

- This is the simplest to replicate outside of DV
- This is because DV really uses audit or an internal version of it for Privilege Analysis
- We can use audit to establish what privileges are used
- We need to analyse the context first
  - If Roles – list all rights per role
  - If context – list all rights for the context
  - Use a version of find_all_privs that creates a row of data for each right
- Enable audit for all rights relative to the context
  - Generate audit commands from the table or policy for PFCLATK
- Create two views (used, unused) based on the audit trail and also the privileges stored and also the context
- Or do a paper based review of audit vs find_all_privs.sql
Conclusions

• Good security design is needed from the start
• Good lock down is needed from the start
• Don’t use SYSTEM ANY
  • Don’t use SYS, SYSTEM and DBA
  • Make changes via proxy to the schema
  • Do not allow DBAs to look at data
• Database Vault is Duct Tape if you do not take care to lock down and secure your data first
• Even if you use DV it must be added on top of good secure design
• So we MUST ALWAYS DESIGN SECURITY FIRST before using additional tools such as DV or not with SE
• DV is built-in so harder to bypass
Questions?

Any Final Questions?
You Don’t Have Database Vault

So, What Can You Do Instead?