



12c Real Application Security

Concepts, Techniques, Use Cases

Dr. Günter Unbescheid
Database Consult GmbH, Jachenau

Database Consult GmbH

- Founded in 1996
- Specialized in ORACLE-based Systems
- Focus Areas
 - Security, Identity Management
 - Tuning, Installation, Configuration, Systemanalysis
 - Support, Troubleshooting, DBA-Tasks
 - Databasesdesign, Datamodelling und –design
 - Custom made Workshops
 - www.database-consult.de
- Since 2012 – Collaboration with



Agenda

- Introduction
- Architecture and Basic Concepts
- Use Case: Step-by-Step Configuration
- Running RAS-enabled Applications
- Auditing RAS Actions
- Summary



Introduction

- RAS – Real Application Security
 - introduced under 12c R1
 - building upon/enhancing concepts like VPD, secure application context, lightweight sessions in OCI, application roles, proxy authentication etc.
- Centralizing security models and authorization logic
 - deploying security policies inside Oracle databases instead of implementation in (distributed) application code
 - avoid inconsistencies by multiple entry access points
 - improve maintenance
- Enforce end-to-end security for 3-tier and 2-tier applications
 - independant of entry points,, usable for:
direct logins, APEX, Java/Middletier application servers
 - enforce least-privilege and preserve client identities



Typical Multitier Access Models

- Application users are not database users
 - authentication and authorization done by application code, bypass(!)
 - connection to database by single (highly-)privileged user
 - identity of end user not known to the database
 - thus intrinsic security features including DB auditing not usable
- Frequent and stateless calls to database
 - connection pool: a cache of database connection objects
 - min and max etc. can be configured, reduce overhead
 - application requests connection from pool – “lightweight sessions”
- Pool implementations
 - JDBC OCI Connection Pool
 - universal Connection Pool (formerly “implicit connection cache”)
 - Database Resident Connection Pool (DRCP)



Typical Database Access Models

- Usually lesser number of users
- „Heavyweight“, longer lasting sessions bound to schema
- Static Privileges via direct or role grants
 - Exception EUS – Enterprise and Global Roles
- Some improvements in former releases
 - client identifier to tag enduser identity by utilizing application context
 - do not confuse with client_info!
 - dynamically enabling privileges through secure application roles
 - enabled through associated PL/SQL Code
 - Enable users to connect through proxy users
 - both have to be configured in the database
 - Use shared schemas and enterprise roles to attach users and privileges
 - involves OID/ODD infrastructure



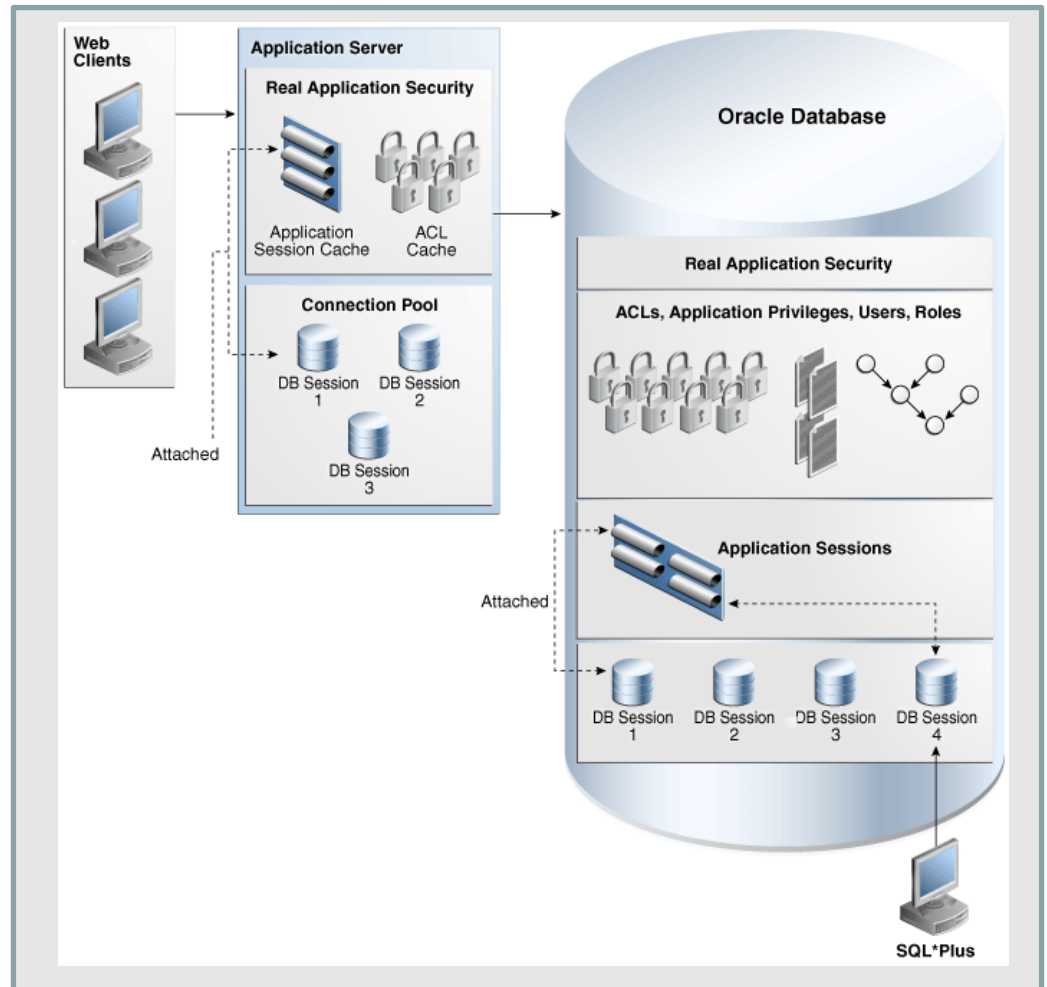
Real Application Security

- Fills the gap by utilizing basic techniques like VPD etc. and by adding an enhanced security framework
- Implemented in 12c Enterprise Edition – no options required
 - 32 dictionary tables under SYS named **XS\$%**
 - Views: 30 **DBA_XS%** and 13 **USER_XS%**
 - 4 predefined database roles named **XS_%**
 - PL/SQL Packages: 18 **XS_%** and 17 **DBMS_XS_%**
 - predefined application user and application roles
 - Java Packages **oracle.security.xs** and **oracle.security.xs.ee.session**
 - Apex application **rasadm** and demo scripts for testing
- Adds new concepts
 - application user, application role, application sessions, ACL, security class, security policy



RAS Architecture

- Connect as
 - direct login application user
 - attach application user
 - switch application user
- Connect from
 - middle tier/connection pool
 - tool such as SQL*Plus



Basic Concepts

- Schema-less application users (AU)
 - do not own any database objects or resources
 - can be but must not be tied to a database schema for name resolution only
 - passwords possible including password rules (profiles)
 - other features like start- and enddates, proxy rules etc.
- Application roles (AR) – group application privileges (AP)
 - are mapped via access control entries (ACE) to application privileges
 - can be granted to AU but not to database roles or users
 - but: database roles can be granted to AR
- Access control lists (ACL)
 - contain ACEs which bind AP to AR
- Data realms (DR) – a logical set of rows in a table used in Sec. Pol.

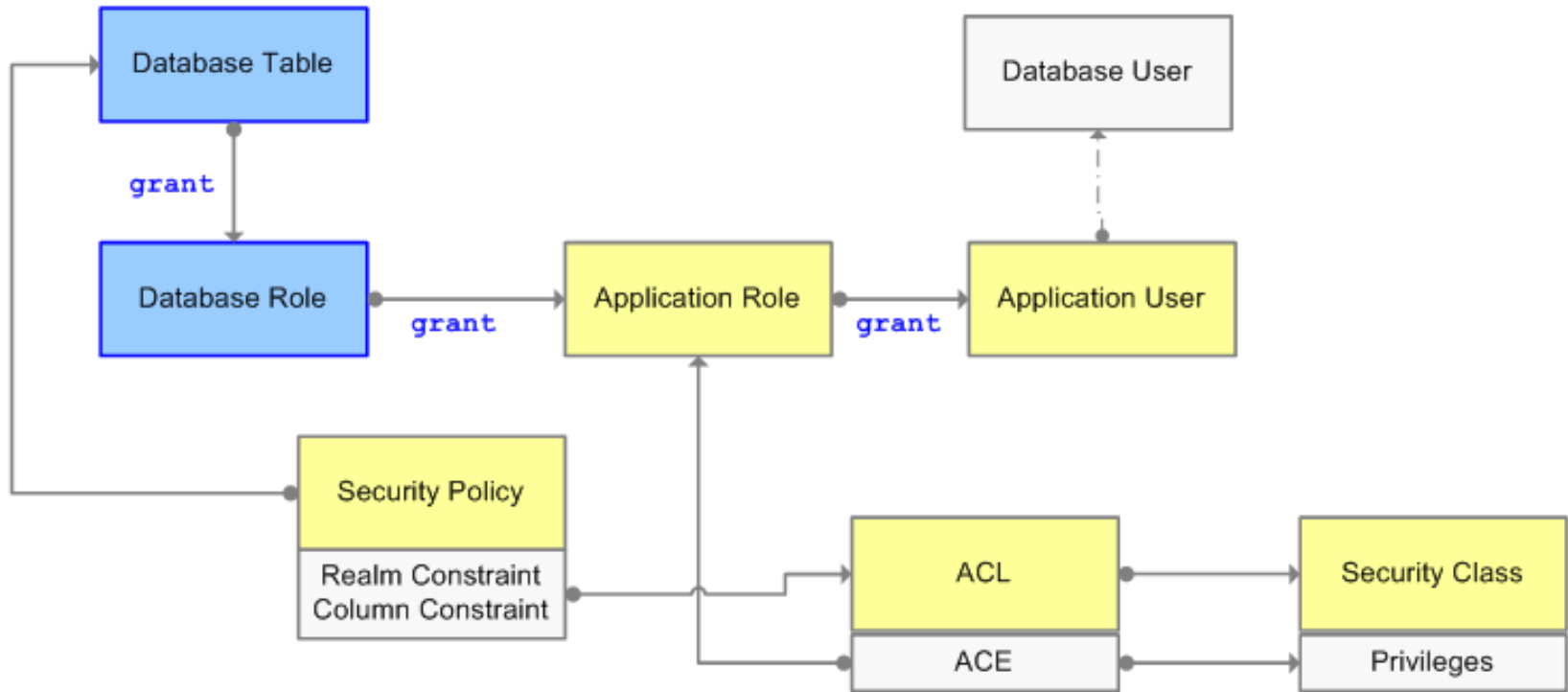


Basic Concepts

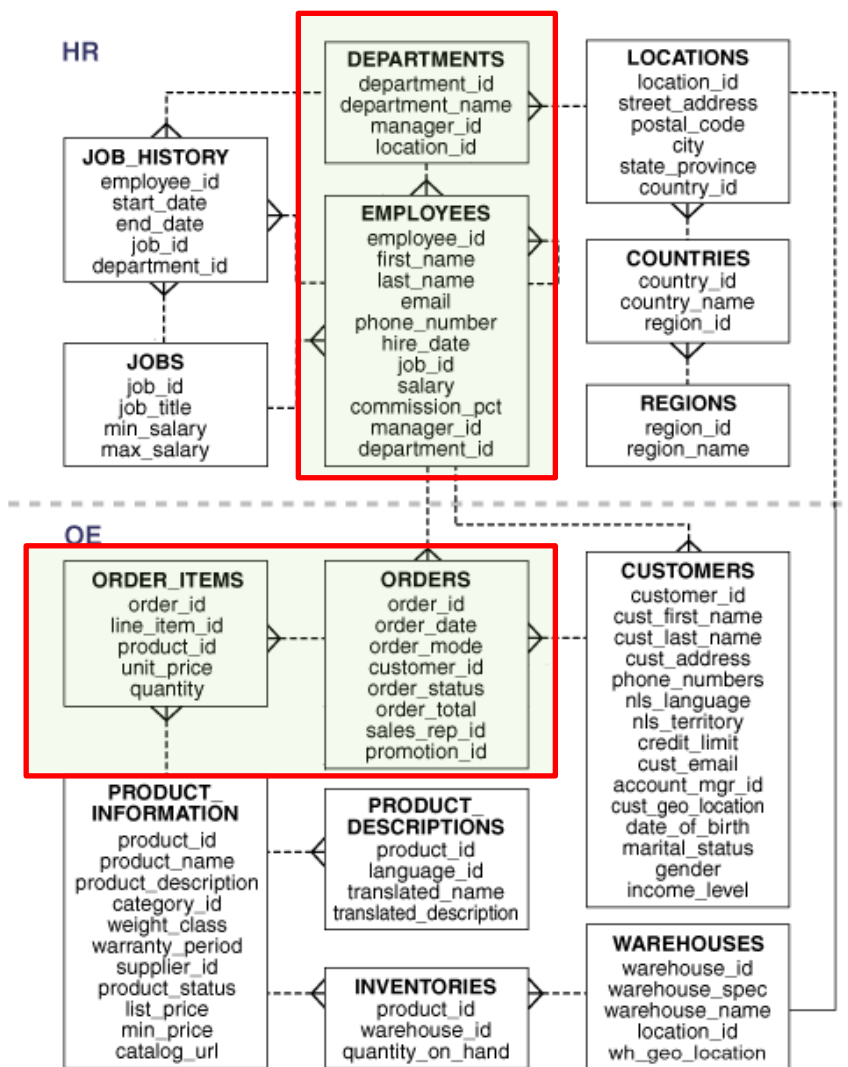
- Application Privileges (AP)
 - control application-level operations on specific business-objects
 - are defined in the context of security classes (SC)
 - can be freely named, predefined AP and SC are available
- Security Policies (SP)
 - SP contain data realms and associate them with ACLs
 - also support column-level authorization to mask column values, in combination with realms they offer cell-level protection
 - once applied on a database table or view, the SP is enforced on all SQL statements/all access paths to the object (2-tier or 3-tier applications)
- Application Session (AS)
 - lightweight, maintain only the security-relevant state for the use
 - multiplexed on heavyweight database sessions among many end-users
 - 3-tier: session acquired from pool and RAS session is attached to connection



Relations (Example)



Building a RAS environment



■ Demo Description

- Employee Oliver Tuvault works in Sales Department and can view:
 - all employees of sales except salary and commission
 - his own record including salary and commission
 - his own orders and order items
- Manager John Russel is manager of Sales
 - can view all employee data and all order data



Step 1: Create RAS Administrator

```
-- e.g. create a database user as RAS administrator  
GRANT dba, xs_session_admin TO rasadm  
IDENTIFIED BY rasadm;
```

- Predefined database role XS_SESSION_ADMIN
 - granted to application role XSSESSIONADMIN which has
 - the ADMINISTER_SESSION application privilege of acl SESSIONACL of security class SESSION_SC
 - contains execute on DBMS_XS_SIDP predefined RAS package
- Check via dictionary views `DBA_ROLES`, `DBA_TAB_PRIVS`, `DBA_XS_ACES`, `DBA_XS_ACLS`, `DBA_XS_ROLE_GRANTS`



Step 2: Create Database Roles

```
-- e.g. under new RAS administrator
-- DATABASE Role
CREATE ROLE db_rasdemo;
GRANT SELECT, INSERT, UPDATE, DELETE
  ON hr.employees TO db_rasdemo;
GRANT SELECT, INSERT, UPDATE, DELETE
  ON hr.departments TO db_rasdemo;
GRANT SELECT, INSERT, UPDATE, DELETE
  ON oe.orders TO db_rasdemo;
GRANT SELECT, INSERT, UPDATE, DELETE
  ON oe.order_items TO db_rasdemo;
```

- role contains maximum object privileges but is not granted to (database) user(s)
- schema OE has synonyms for HR objects



Step 3: Create Application Roles

```
-- Create an application role sales_emp for employees of sales dep.  
exec xs_principal.create_role(name => 'sales_emp', enabled  
=> true, description => 'for employees of sales');  
  
-- Create an application role sales_mgr for managers of sales department  
exec xs_principal.create_role(name => 'sales_mgr', enabled  
=> true, description => 'for managers of sales');  
  
-- grant database role to application roles  
GRANT db_rasdemo TO sales_emp;  
GRANT db_rasdemo TO sales_mgr;
```

- Check via views DBA_XS_ROLES and DBA_XS_ROLE_GRANTS
- Application roles are not granted to database users



Step 4: Create Application Users

```
-- Create application user OTUVAULT (Oliver Tuvault, sales rep.)
exec xs_principal.create_user(name => 'otuvault',
    schema => 'oe');
exec xs_principal.set_password('otuvault', 'welcome1');
exec xs_principal.grant_roles('otuvault', 'sales_emp');
-- Create application user JRUSSEL (John Russell, sales mgr.)
exec xs_principal.create_user(name => 'jrussel',
    schema => 'oe');
exec xs_principal.set_password('jrussel', 'welcome1');
exec xs_principal.grant_roles('jrussel', 'sales_mgr');
```

- Application user do not own database schemas
- `CURRENT_SCHEMA` set to OE
- Users can „attach“ session or directly connect to database
- If required password profile and start-/enddates can be set



Step 4: Application Users

```

SQL> conn otuvault/welcome1
Connected.
SQL> sho user
USER is "OTUVAULT"
SQL> select user from dual;
USER
-----
XS$NULL

select sys_context('userenv','current_schema') cs
       , sys_context('userenv','current_user') cu
       , xs_sys_context('xs$session','username') xsu from dual;
CS          CU          XSU
-----
OE          XS$NULL    OTUVAULT

select role from session_roles;
ROLE
-----
DB_RASDEMO

```



Step 4: Application Users

```
SQL> select 'AppRole' origin, role_name role
from v$xs_session_roles union
select 'DbRole' origin, role from session_roles order by 1;
```

```
ORIGIN  ROLE
-----  -----
AppRole DBMS_AUTH
AppRole DBMS_PASSWD
AppRole SALES_EMP
AppRole XSAUTHENTICATED
AppRole XSPUBLIC
DbRole  DB_RASDEMO
```



Step 5: Create Security Class

```
DECLARE
  pr_list XS$PRIVILEGE_LIST;
BEGIN
  pr_list :=XS$PRIVILEGE_LIST(
    XS$PRIVILEGE(name=>'VIEW_SENSITIVE_INFO'),
    XS$PRIVILEGE(name=>'UPDATE_INFO',
      implied_priv_list=>XS$NAME_LIST
        ('"UPDATE"', '"DELETE"', '"INSERT"')));
  xs_security_class.create_security_class(
    name => 'sales_privileges',
    parent_list => xs$name_list('sys.dml'),
    priv_list => pr_list);
END;
```

- Define a set of application privileges for use in ACLs



Step 6: Create ACLs

```
-- create ACLs referring privileges and application roles
DECLARE
  aces xs$ace_list := xs$ace_list();
Begin
  aces.extend(1);
  aces(1) := xs$ace_type(privilege_list =>
xs$name_list('select', 'VIEW_SENSITIVE_INFO'),
  principal_name => 'sales_emp');
xs_acl.create_acl(name => 'emp_acl'
, ace_list => aces
, sec_class => 'sales_privileges');
END;
```

- Binding SELECT and VIEW_SENSITIVE_INFO application privileges to application role SALES_EMP by means of ACL named EMP_ACL
- ... more ACLs are necessary for our demo ...
- Check via [DBA_XS_ACLS](#) and [DBA_XS_ACES](#) views



Step 7: Realms and Data Security Policies

```
declare
  realms    xs$realm_constraint_list := xs$realm_constraint_list();
  cols      xs$column_constraint_list := xs$column_constraint_list();
begin
  realms.extend(1);
  realms(1) := xs$realm_constraint_type(
    realm => 'email = xs_sys_context(''xs$session'', ''username'')',
    acl_list => xs$name_list('emp_acl'));

  cols.extend(1);
  cols(1) := xs$column_constraint_type(
    column_list => xs$list('salary', 'commission_pct')
    , privilege   => 'VIEW_SENSITIVE_INFO');

  xs_data_security.create_policy(
    name           => 'salesemp_ds',
    realm_constraint_list => realms, column_constraint_list => cols);
end;
```

- Row constraint (data realm) – view only ones own record
- Column constraint – exclude sensitive columns when no privilege granted



Step 7: Using ACL Parameter

```
...
realms(2) := xs$realm_constraint_type(
    realm => 'manager_id = &' || 'MGR');
...
xs_data_security.create_policy( ...
xs_data_security.create_acl_parameter( policy =>
'salesemp_ds', parameter => 'MGR',
    param_type => XS_ACL.TYPE_NUMBER);
...
BEGIN
    sys.xs_acl.add_acl_parameter(acl => 'mgr_acl',
    policy => 'salesemp_ds', parameter => 'MGR',
    value => 145);
END;
```



Step 8: Apply Security Policy

```
begin
  xs_data_security.apply_object_policy(
    policy => 'salesemp_ds',
    schema => 'hr',
    object => 'employees',
    owner_bypass => true);
end;
```

- Bind policy to database table or view
- Once applied enabled for ALL accesses except for
 - `owner_bypass => true` (default is false(!))
 - system privilege **EXEMPT ACCESS POLICY**



Validation

- The complex structure makes it necessary to validate thoroughly
- **VALIDATE_PRINCIPAL** Function
- **VALIDATE_SECURITY_CLASS** Function
- **VALIDATE_ACL** Function
- **VALIDATE_DATA_SECURITY** Function
 - Validates the data security policy or validates the data security policy against a specific table.
- **VALIDATE_NAMESPACE_TEMPLATE** Function
 - Validates the namespace template.
- **VALIDATE_WORKSPACE** Function
 - Validates an entire workspace



Work with RAS

- Direct login with application user
 - password has to be provided, usual syntax
- Attach application user to existing session
 - starting from low privilege user/session

```
EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE (
  'ADMINISTER_SESSION', 'RASDUMMY', XS_ADMIN_UTIL.PTYPE_DB);

-- db user with only create session and administer session privs
connect rasdummy/rasdummy
-- attach
DECLARE
  SESSIONID RAW(16);
BEGIN
  SYS.DBMS_XS_SESSIONS.CREATE_SESSION('JRUSSEL', sessionid);
  SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
END;
/

select xs_sys_context('XS$SESSION', 'CURRENT_XS_USER') from dual;
```



Middle Tier

- Mid-Tier connections in dispatcher mode, e.g.
 - Special application user **dispatcher** with session and cache privileges
 - dispatcher handles connection (pool) as direct login user (trusted)
 - application user without session and cache privileges
- **XSSessionManager** class manages the life cycle of the session.
 - methods to create, attach, assign, detach, and destroy sessions
 - e.g. method **getSessionManager** gets an instance of Session M.
- Application Session handling
 - create session (**XSSessionManager** method **createSession**)
 - attach session (with app user) to traditional db session (method **attachSession**)
 - assign application user to attached anonymous (XSGUEST) application session
 - switch security context of current application session for another app.u.



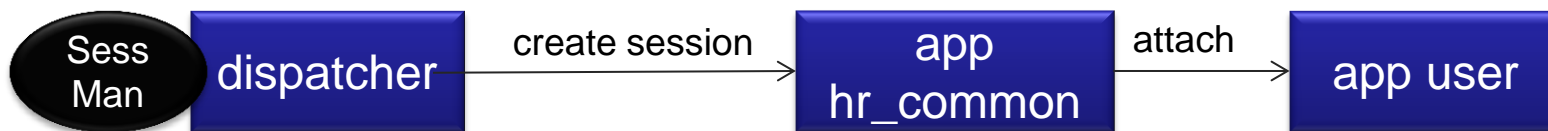
Middle Tier (Code Snippets)

```
// dispatcher's connection
mgrConnection = DriverManager.getConnection(url, "dispatcher", "welcome1");
// RAS session manager
manager = XSSessionManager.getSessionManager(mgrConnection, 30, 2048000);
// connection used for application query
appConnection = DriverManager.getConnection(url, "hr_common", "hr_common");
```

```
private void queryAsUser(String user) throws SQLException ...

Session lws = manager.createSession(appConnection, user, null,null);

manager.attachSession(appConnection, lws, null, null, null, null, null);
```



RAS Admin

ORACLE RAS Administration admin Help Logout

Home Policies Privileges Namespaces Users Roles Settings

Home

Introduction

Policy Summary

Users and Roles		Count	Data Security		Count
Application Users from External Stores		0	Data Security Policies		1
Application Users in Database		8	Data Realms		2
Application Roles from External Stores		0	Privilege Classes		3
Application Roles in Database		8	Application Namespaces		0
		1 - 4			1 - 4

Policy Modification Report

Modification Count	Last 1 Hour	Last 24 Hours	Last 7 Days	Last 30 Days
Data Security Policies	0	1	1	1
Privilege Grants to Data Realms	0	0	0	0
Application Users in Database	0	1	3	9
Application Roles in Database	0	0	2	20
1 - 4				

- Apex 5.0 Application, free download
- Basic Admin Pages for viewing, adding and deleting elements



Auditing

- RAS actions audited only by unified audit policies
 - Standard Audit sees only connect under generic XS\$NULL, no RAS actions
- Predefined unified audit policies available (COMPONENT=XS)
 - ORA_RAS_POLICY_MGMT
 - administrative actions on application users, roles, and policies
 - ORA_RAS_SESSION_MGMT
 - run-time Oracle Real Application Security session actions and namespace actions
- Particular View fro XS-Actions
 - DBA_XS_AUDIT_POLICY_OPTIONS
 - DBA_XS_ENB_AUDIT_POLICIES – enabled policies
 - DBA_XS_AUDIT_TRAIL – aucit actions



Summary

- Meant for high level applications with (serious) security concerns and manifold user population
- Many options to centralize security policies for applications
- (Very) complex configuration, allow learning curve
- detailed documentation, many examples
 - external user deployment badly documented
 - so far no export/import mechanisms
- Partly new technology, just a few if any blogs so far
 - interesting for APEX applications
- No experiences as far as stability in the long run is concerned
- Gather experience in test cases



Thank you for listening
www.database-consult.de

