Un exemple complet d'implémentation d'*Oracle Real Application Security* avec Oracle APEX

Contenu Afficher

Objectif

Objectif: établir une claire séparation des rôles entre l'équipe de développement et l'équipe chargée du modèle d'autorisations applicatives.

Plutôt que d'ajouter des filtres au niveau de l'application APEX ou bien au niveau de vues, l'approche est, ici, d'utiliser les fonctionnalités de Oracle RAS afin de mettre en place un contrôle d'accès via des rôles applicatifs.

Scénario

Une société fictive, *BioAdvice*, se prépare à lancer le *Biogreen*, un pesticide basé sur une nouvelle molécule.
Pour anticiper le volume à produire, l'équipe commerciale est chargée de sonder sa base installée afin d'enregistrer les intentions de commandes.
La force de vente de *BioAdvice* est organisée en cinq régions, avec un responsable régional à la tête et quelques commerciaux dans chacune des régions. Un directeur commercial supervise le niveau national et il est aidé par un assistant commercial. (voir *Organisation* en annexe).

Les règles d'accès et de visibilité sont les suivantes:

- Le directeur commercial a accès sans restriction à toutes les informations.
- L'assistant a accès à toutes les données, excepté la valeur du champs DISCOUNT

- Le responsable régional a accès à toutes les données relative à la région dont il a la responsabilité.
- Le commercial ne voit que les données, sans restriction, des clients qui sont dans son secteur.

Pour des raisons de commodité, un tableau excel a été envoyé à chaque commercial pour que celui-ci y enregistre les pré-commandes. L'idée est de consolider les tableaux qui seront remontés au fil de l'eau.

Les tableaux sont remontés par mail ou déposés sur un *file system* puis assemblés via une macro excel.

Après quelques semaines de fonctionnement, cette approche s'avère inopérante car il est impossible d'obtenir une valeur consolidée en temps réel avec un bon niveau de fiabilité. On se retourne alors vers une solution basée sur un base de données unique avec un accès en ligne via une application web. La plateforme de développement retenue est Oracle APEX.

Une application est rapidement générée et l'échantillon ci-dessous est montré aux utilisateurs qui adhèrent à cette nouvelle interface.

La question qui reste à traiter concerne la visibilité des données. Comment respecter les règles qui ont été mentionnées plus haut ?
Lorsqu'on diffuse un tableau spécifique à chaque commercial, le problème est réglé par le fait même que le travail de découpage est déjà réalisé sous la forme de x tableaux spécifiques. Dans le cas d'une base centralisée, il faut ajouter une notion d'utilisateur et de droits d'accès. C'est ici qu'intervient *Real Application Security*

(RAS).

Précisons que l'emploi de RAS n'est nullement un passage obligé ! On peut envisager une solution simple (mais présentant quelques défauts comme il sera discuté plus loin) consistant à créer une vue jouant le rôle de filtre.

Cette vue sera du type :

```
Rename ADO_PLAN to ADO_PLAN_T;
create or replace view ADO_PLAN as
 select * from ADO_PLAN_T
 where region in (select region
             from ado_emp
             where job = 'responsable régional' and
             login = lower(v('APP_USER'))
                 )
 or
 exists (select 1
     from ado_emp
     where login =lower(v('APP_USER')) and
     job = 'directeur commercial'
     )
 or
 commercial = lower(v('APP_USER'))
```

On met à profit la connaissance de l'utilisateur qui est authentifié dans l'application APEX pour utiliser cette valeur ('APP_USER') dans le filtre de la vue. L'application reste inchangée. Il y a cependant une règle de sécurité qui devra être résolue en programmant directement une condition au niveau du champs DISCOUNT, dans la page liée au formulaire de détail ainsi que dans le rapport sur les commandes.

Génération d’une application APEX

Mise en oeuvre de RAS

Prise en compte de RAS dans l’interface APEX

Les utilisateurs doivent être enregistrés à la fois dans Oracle APEX, en tant qu’utilisateur et dans RAS en tant que *principal*. L’authentification sera toujours sous le contrôle d’APEX. A ce propos, un mécanisme de provisionnement est à mettre en place pour des architecture s’appuyant massivement sur RAS.

Masquage de colonnes pour certains rôles

RAS permet de masquer des colonnes. Dans notre exemple, les assistants commerciaux peuvent visualiser toutes les commandes, mais ne peuvent pas avoir connaissance du taux de discount accordé par le commercial pour son client. Observer que la colonne DISCOUNT est affichée avec des ‘xxx’.
On utilise la fonction:

```
COLUMN_AUTH_INDICATOR(col)
RETURN BOOLEAN;
```

Limitations, anomalies, bugs

Plugins

A ce stade, les tests que j'ai réalisés avec un plugin sont négatifs. C'est à dire que une requête sur la table DEMO.ADO_PLAN à l'intérieur d'un plugin ne ramène aucune ligne. Cela est bloquant dans l'usage d'un plugin de visualisation des données sur une maps Google que j'avais prévu d'utiliser.

Latence dans l'activation de RAS
Lors de l'activation de RAS, un délai de plusieurs heures peut s'avérer nécessaire pour que l'authentification fonctionne. Ci-dessous le message d'erreur obtenu. Quelques heures plus tard, ce message n'apparaissait plus.

Usage d'une *subquery* dans la définition du *Data Policy*
Je n'ai pas réussi a utiliser une sous-requête en voulant exprimer la règle concernant la région pour les responsables régionaux. Je pensais utiliser au départ une table telle qu'elle figure en annexe et invoquer le prédicat suivant:

```
'REGION IN (select REGION from DEMO.ADO_EMP where REGION = '||'REGION
and -- login = lower(xs_sys_context(''xs$session'',''username'')))
dans la définition du REALM associé à l'ACL portant sur les
responsables régionaux.
(à la place de
       realms(2) := xs$realm_constraint_type(
                           realm => 'REGION = &' || 'PREGION'
                     );
)
```

mais j'ai toujours abouti à des erreurs sur le prédicat. J'ai donc laissé tomber cette piste et j'ai créé autant d'ACLs qu'il y avait de régions. D'un point de vue « philosophique » c'est peut-être la meilleure approche dans la mesure où les

informations sur l'organisation sont déportées dans le dictionnaire de RAS et non pas dans une table (ADO_EMP) d'un schéma applicatif.

Annexes

Scripts de création des *Principals*, *ACLs* et *Data Security Policies*

Tous les scripts de création d'un modèle d'autorisation, ci-dessous, sont accessibles dans leur dernière version sur github.

Principals

```
exec xs_principal.create_role(name => 'sr_role', enabled => true);
exec xs_principal.create_role(name => 'hq_role', enabled => true);
exec xs_principal.create_role(name => 'sa_role', enabled => true);
exec sys.xs_principal.create_role(name => 'sud_ventes', enabled =>
TRUE);
exec sys.xs_principal.create_role(name => 'nord_ventes', enabled =>
TRUE);
exec sys.xs_principal.create_role(name => 'est_ventes', enabled =>
TRUE);
exec sys.xs_principal.create_role(name => 'ouest_ventes', enabled =>
TRUE);
exec sys.xs_principal.create_role(name => 'centre_ventes', enabled =>
TRUE);
 grant db_bio to sr_role;
 grant db_bio to hq_role;
 grant db_bio to sa_role;
 grant db_bio to ouest_ventes;
 grant db_bio to sud_ventes;
 grant db_bio to nord_ventes;
```

```
grant db_bio to est_ventes;
grant db_bio to centre_ventes;
-- Directeur ventes
exec  xs_principal.create_user(name => 'alain', schema => 'DEMO');
exec  sys.xs_principal.set_password('alain', 'alain');
exec  xs_principal.grant_roles('alain', 'hq_role');
-- Responsable régional
exec  xs_principal.create_user(name => 'ygor', schema => 'DEMO');
exec  sys.xs_principal.set_password('ygor', 'ygor');
exec  xs_principal.grant_roles('ygor', 'ouest_ventes');
-- Commercial
exec  xs_principal.create_user(name => 'brice', schema => 'DEMO');
exec  sys.xs_principal.set_password('brice', 'brice');
exec  xs_principal.grant_roles('brice', 'sr_role');
-- Assistant commercial
exec  xs_principal.create_user(name => 'pierre', schema => 'DEMO');
exec  sys.xs_principal.set_password('pierre', 'pierre');
exec  xs_principal.grant_roles('pierre', 'sa_role');

Give create session privilege
BEGIN
    SYS.XS_PRINCIPAL.GRANT_ROLES('ALAIN', 'XSCONNECT');
    SYS.XS_PRINCIPAL.GRANT_ROLES('YGOR', 'XSCONNECT');
    SYS.XS_PRINCIPAL.GRANT_ROLES('BRICE', 'XSCONNECT');
    SYS.XS_PRINCIPAL.GRANT_ROLES('PIERRE', 'XSCONNECT');
END;
/
```

ACLs

```
Creating ACLs: EMP_ACL, IT_ACL, and HR_ACL
(Ace stands for Access Control Entry)
```

```
 declare
       aces xs$ace_list := xs$ace_list();
     begin
       aces.extend(1);
 -- SR_ACL: This ACL grants SR_ROLE the privileges to view a sales
rep's
 --          own record including DISCOUNT column.
 aces(1) := xs$ace_type(
             privilege_list => xs$name_list('select','view_discount'),
             principal_name => 'sr_role');
sys.xs_acl.create_acl(name => 'sr_acl',
ace_list  => aces,
sec_class => 'bioprivs');
 -- HQ_ACL:  This ACL grants HQ_ROLE the privileges to view and update
all
 -- order records including DISCOUNT column.
 aces(1):= xs$ace_type(
          privilege_list => xs$name_list('all'),
          principal_name => 'hq_role');
sys.xs_acl.create_acl(name => 'hq_acl',
ace_list  => aces,
sec_class => 'bioprivs');
 -- SA_ACL:  This ACL grants SA_ROLE the privileges to view and update
all
 -- order records excluding DISCOUNT column.
      aces(1):= xs$ace_type(
          privilege_list => xs$name_list('select'),
          principal_name => 'sa_role');
 sys.xs_acl.create_acl(name => 'sa_acl',
ace_list  => aces,
sec_class => 'bioprivs');
 end;
```

```
    /
```

Data Security Policy

```
 --Example 5-19 Creating the EMPLOYEES_DS Data Security Policy
 declare
    realms   xs$realm_constraint_list := xs$realm_constraint_list();
    cols     xs$column_constraint_list := xs$column_constraint_list();
 begin
    realms.extend(4);
 -- Realm #1: Only the order's own record.
 --  SR_ROLE can view the realm including DISCOUNT column.
    realms(1) := xs$realm_constraint_type(realm => 'COMMERCIAL =
lower(xs_sys_context(''xs$session'',''username''))',
 acl_list => xs$name_list('sr_acl'));

 -- Realm #2: The records in the same region as the Region Manager.  -
-                 realms(2) := xs$realm_constraint_type(realm =>
'REGION = &' || 'PREGION' );
 -- Realm #3: All the records.
 -- HQ_ROLE can view and update the realm including DISCOUNT column.
realms(3) := xs$realm_constraint_type(realm => '1 = 1',
   acl_list => xs$name_list('hq_acl'));

  --  Realm #4R: All the records and no acces to DISCOUNT.
  --  SA_ROLE can view and update the realm excluding DISCOUNT column.
   realms(4) := xs$realm_constraint_type(
        realm    => '1 = 1',
        acl_list => xs$name_list('sa_acl'));
-- Column constraint protects DISCOUNT column by requiring
view_discount  -- privilege.
 cols.extend(1);
```

```
 cols(1) := xs$column_constraint_type(column_list =>
xs$list('DISCOUNT'),
     privilege   => 'view_discount');
sys.xs_data_security.create_policy(name=> 'bio_ds',
  realm_constraint_list  => realms,
  column_constraint_list => cols);
sys.xs_data_security.create_acl_parameter(policy => 'bio_ds',
  parameter => 'PREGION',
  param_type => XS_ACL.TYPE_VARCHAR);
end;
/
```

ACL pour une région

```
DECLARE
   ace_list XS$ACE_LIST;
 BEGIN
   ace_list := XS$ACE_LIST(
              XS$ACE_TYPE(privilege_list => XS$NAME_LIST('SELECT'),
                      granted => true,
                      principal_name => 'ouest_ventes'),
              XS$ACE_TYPE(privilege_list => XS$NAME_LIST('SELECT',
'view_discount'),
            granted => true,
           principal_name => 'ouest_ventes'));
 sys.xs_acl.create_acl(name => 'view_ouest_ventes',
                ace_list => ace_list,
                sec_class => 'bioprivs',
                description => 'Authorize read access for the ouest
region');
 sys.xs_acl.add_acl_parameter(acl => 'view_ouest_ventes',
                           policy => 'bio_ds',
```

```
                              parameter => 'PREGION',
                              value => 'ouest');
 END;
 /
 -- ACLs for other regions have to be done !
```

Validation

```
-- Validating policy
 begin
   if (sys.xs_diag.validate_workspace()) then
     dbms_output.put_line('All configurations are correct.');
   else
     dbms_output.put_line('Some configurations are incorrect.');
   end if;
 end;
 /
```

Activation

```
-- Apply the data security policy to the EMPLOYEES table.
 begin
   xs_data_security.apply_object_policy(
     policy => 'bio_ds',
     schema => 'DEMO',
     object =>'ADO_PLAN');
 end;
 /
```

Jeu de test

Organisation

Le tableau ci-dessous récapitule les *Business Roles* dans l'organisation de la société *BioAdvice*. Chacun sera traduit en un ou plusieurs rôles applicatifs. Les noms en gras sont ceux échantillonnés pour la démo

NOM	PRENOM	LOGIN	REGION	JOB
Klaesson	Filmore	filmore	centre	assistant commercial
Ginnety	Adrien	adrien	centre	responsable régional
Costes	Alain	alain	corp	directeur commercial
Grinish	Taite	taite	corp	directeur marketing
Early	Merrick	merrick	est	commercial
Mc Pake	Waverley	waverley	est	commercial
Goding	Charles	charles	est	responsable régional
Kettel	Brennen	brennen	nord	commercial
Ianno	Adam	adam	nord	responsable régional
Petchey	Cory	cory	nord	commercial
Creany	Ivar	ivar	ouest	commercial
Calkin	Ygor	ygor	ouest	responsable régional
Tchaikov	Patrick	patrick	ouest	commercial
Spondley	Talbert	talbert	ouest	commercial
Duguet	Pierre	pierre	ouest	assistant commercial
Lacombes	Brice	brice	ouest	commercial
Fattore	Everett	everett	sud	commercial
Randall	Cleon	cleon	sud	responsable régional
Stedall	Garrot	garrot	sud	commercial

Feuille excel des commandes consolidées

Ce tableau est celui utilisé pour l'initialisation des données dans la table ADO_PLAN. Il

est le résultat de la consolidation de tous les tableaux partiels remontés par les commerciaux.

Author

Patrick

GPM Factory