Role-Based Access Control

Overview

- Role Hierarchy (RH)
- User Assignment (UA)
- Permission Assignment (PA)
- Users (USERS)
- Roles (ROLES)
- Sessions (SESSIONS)
- OPS
- OBS
- PRMS

Diagram:
- SSD
- (RH) Role Hierarchy
- (UA) User Assignment
- (PA) Permission Assignment
- user_sessions
- session_roles
- DSD
Objective

- Establish a common vocabulary for Role-Based Access Control for use in SEPM
- Present a Framework for Role-Based Access Control for both Physical and Virtual Domains
- Discuss Various AC Models and why RBAC is a must!!!!
Think about this…

- “Although the fundamental concepts of roles are common knowledge, the capability to formalize model specifications needed to implement RBAC models is beyond the knowledge base of existing staff in many software companies”

- “The lack of knowledge and staff expertise in the area of RBAC increases the uncertainty of both the technical feasibility of developing successful RBAC-enabled products and the development cost and time frame.”

*The Economic Impact of Role Based Access Control*
Access Controls Types

- Discretionary Access Control
- Mandatory Access Control
- Role-Based Access Control
Discretionary AC

- Restricts access to objects based solely on the identity of users who are trying to access them.

<table>
<thead>
<tr>
<th>Application Access List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Tom</td>
</tr>
<tr>
<td>John</td>
</tr>
<tr>
<td>Cindy</td>
</tr>
</tbody>
</table>
Mandatory AC

- MAC mechanisms assign a security level to all information, assign a security clearance to each user, and ensure that all users only have access to that data for which they have a clearance.

Principle: Read Down Access
- equal or less Clearance

Write Up Access
- equal or higher Clearance

Better security than DAC
Mandatory AC (cont)

Individuals

Resources

Server 1
“Top Secret”

Server 2
“Secret”

Server 3
“Classified”
Role-Based AC

- A user has access to an object based on the assigned role.
- Roles are defined based on job functions.
- Permissions are defined based on job authority and responsibilities within a job function.
- Operations on an object are invoked based on the permissions.
- The object is concerned with the user’s role and not the user.
Role-Based AC

User’s change frequently, Roles don’t
Privilege

- Roles are engineered based on the principle of least privileged.
- A role contains the minimum amount of permissions to instantiate an object.
- A user is assigned to a role that allows him or her to perform only what’s required for that role.
- No single role is given more permission than the same role for another user.
Role-Based AC Framework

- Core Components
- Constraining Components
  - Hierarchical RBAC
    - General
    - Limited
  - Separation of Duty Relations
    - Static
    - Dynamic
Core Components

- Defines:
  - USERS
  - ROLES
  - OPERATIONS (ops)
  - OBJECTS (obs)
  - User Assignments (ua)
    - assigned_users
Core Components (cont)

- Permissions (*prms*)
  - Assigned Permissions
  - Object Permissions
  - Operation Permissions

- Sessions
  - User Sessions
  - Available Session Permissions
  - Session Roles
Constraint Components

- Role Hierarchies ($rh$)
  - General
  - Limited
- Separation of Duties
  - Static
  - Dynamic
# RBAC Transition

<table>
<thead>
<tr>
<th>Models</th>
<th>Hierarchies</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBAC_0</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RBAC_1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>RBAC_2</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>RBAC_3</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Least Privileged Separation of Duties*

*Most Complex*

[Diagram showing RBAC model transition with effort and RBAC model on the x and y axes respectively.]
RBAC System and Administrative Functional Specification

- Administrative Operations
  - Create, Delete, Maintain elements and relations

- Administrative Reviews
  - Query operations

- System Level Functions
  - Creation of user sessions
  - Role activation/deactivation
  - Constraint enforcement
  - Access Decision Calculation
Core RBAC
USERS

Person

Intelligent Agent

Proces
ROLES

An organizational job function with a clear definition of inherent responsibility and authority (permissions).

Developer

Budget Manager

Help Desk Representative

MTM relation between USERS & PRMS

Director
OPS (operations)

An execution of a program specific function that's invoked by a user.

- Database – Update  Insert  Append  Delete
- Locks – Open  Close
- Reports – Create  View  Print
- Applications - Read  Write  Execute
OBS (objects)

An entity that contains or receives information, or has exhaustible system resources.

- OS Files or Directories
- DB Columns, Rows, Tables, or Views
- Printer
- Disk Space
- Lock Mechanisms

RBAC will deal with all the objects listed in the permissions assigned to roles.
UA (user assignment)

A user can be assigned to one or more roles

A role can be assigned to one or more users

$UA \subseteq \text{USERS} \times \text{ROLES}$
UA (user assignment)

Mapping of role $r$ onto a set of users

ROLES set

User.F1
User.F2
User.F3
User.DB1

- View
- Update
- Append

permissions object

USERS set

assigned_user : (r : ROLES) → 2^users
assigned_user(r) = \{ u ∈ USERS | (u, r) ∈ UA \}
PRMS (permissions)

The set of permissions that each grant the approval to perform an operation on a protected object.

$PRMS = 2^{(OPS \times OBS)}$
PA (prms assignment)

PRMS set

A prms can be assigned to one or more roles

Create
Delete
Drop
View
Update
Append

PA ⊆ PRMSxROLES

ROLES set

A role can be assigned to one or more prms

Admin.DB1

User.DB1
PA (prms assignment)

Mapping of role $r$ onto a set of permissions

ROLES set $\rightarrow$ PRMS set

User.F1
User.F2
User.F3
Admin.DB1

assigned permissions($r : ROLES) \rightarrow 2^{PRMS}$

assigned permissions($r$) = \{ $p \in PRMS \mid (p, r) \in PA$ \}
PA (prms assignment)

Mapping of operations to permissions

OPS set

public int read(byteBuffer dst)
  throws IOException

Inherited methods from java.nio.channels
  close()
  isOpen()

PRMS set

READ

Gives the set of ops associated with the permission

\[ \text{Ob}(p : PRMS) \rightarrow \{op \subseteq OPS\} \]
PA (prms assignment)

Mapping of permissions to objects

PRMS set

• Open
• Close

Objects

BLD1.door2

SQL

DB1.table1

Gives the set of objects associated with the prms

\[ Ob(p : PRMS) \rightarrow \{ ob \subseteq OBS \} \]
SESSIONS

The set of sessions that each user invokes.
The mapping of user $u$ onto a set of sessions.

$\text{user}\_\text{sessions}(u: \text{USERS}) \rightarrow 2^{\text{SESSIONS}}$
The mapping of session $s$ onto a set of roles

$$session\_\text{roles}(s : \text{SESSIONS}) \rightarrow 2^{\text{ROLES}}$$

$$session\_\text{roles}(s) \subseteq \{ r \in \text{ROLES} \mid (session\_\text{users}(s), r \in UA) \}$$
SESSIONS

Permissions available to a user in a session.

\[
\text{avail}_\text{session}_\text{persm}(s:\text{SESSIONS}) \rightarrow 2^{\text{PRMS}}
\]

\[
\bigcup_{r:\text{session}_\text{roles}(s)} \text{assigned}_\text{permissions}(r)
\]
Hierarchical RBAC

- USERS
- ROLES
- SESSIONS
- OPS
- OBS
- PRMS

Role Hierarchy
User Assignment
Permission Assignment
user_sessions
session_roles
Lattice Hierarchy

Director
  / \  
Project Lead 1  Project Lead 2
    / \          / \
  Production  Quality  Production  Quality
Engineer 1  Engineer 1  Engineer 2  Engineer 2
    / \          / \
  Engineer 1  Engineer 2

Engineering Dept
**RH (Role Hierarchies)**

- Natural means of structuring roles to reflect organizational lines of authority and responsibilities
- General and Limited
- Define the inheritance relation among roles
  
  i.e. \( r_1 \) *inherits* \( r_2 \)

  
  User \( r\cdot w\cdot h \)
  
  Guest -r-
General RH

- Guest Role Set
- User Role Set
- Power User Role Set
- Admin Role Set

Support Multiple Inheritance

i.e. \( r_1 \) inherits \( r_2 \)

User
\( r\)-w-h

Guest
\( -r\)-

Only if all users of \( r_1 \) are also users of \( r_2 \)

\( r_1 \rightarrow r_2 \Rightarrow \text{authorized permissions}(r_2) \subseteq \text{authorized permissions}(r_1) \)

\( \land \text{authorized users}(r_1) \subseteq \text{authorized users}(r_2) \)
authorized users

Mapping of a role onto a set of users in the presence of a role hierarchy

ROLES set

Admin.DB1
User.DB1
User.DB1
User.DB1

First Tier USERS set

User.DB1

\[ \text{authorized}_\text{users}(r) = \{ u \in \text{USERS} \mid r' \rightarrow r(u, r') \in UA \} \]
authorized permissions

Mapping of a role onto a set of permissions in the presence of a role hierarchy

ROLES set

User.DB1
User.DB1
User.DB1
Admin.DB1

PRMS set

- View
- Update
- Append
- Create
- Drop

authorized_permissions(r : ROLES) = \{ p \in PRMS \mid r' > r, (p', r') \in PA \}
Limited RH

A restriction on the immediate descendants of the general role hierarchy

\[ \forall r, r_1, r_2 \in ROLES, r > r_1 \land r > r_2 \Rightarrow r_1 = r_2 \]
Notice that Frank has two roles: Billing and Cashier. This requires the union of two distinct roles and prevents Frank from being a node to others.
Constrained RBAC
Separation of Duties

- Enforces conflict of interest policies employed to prevent users from exceeding a reasonable level of authority for their position.
- Ensures that failures of omission or commission within an organization can be caused only as a result of collusion among individuals.
- Two Types:
  - Static Separation of Duties (SSD)
  - Dynamic Separation of Duties (DSD)
SSD

- SSD places restrictions on the set of roles and in particular on their ability to form UA relations.
- No user is assigned to \( n \) or more roles from the same role set, where \( n \) or more roles conflict with each other.
- A user may be in one role, but not in another—mutually exclusive.
- Prevents a person from submitting and approving their own request.

\[
SSD \subseteq (2^{\text{ROLES}} \times \mathbb{N})
\]

\[
\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} \text{assigned}_r \text{users}(r) = \emptyset
\]
SSD in Presence of RH

- A constraint on the authorized users of the roles that have an SSD relation.
- Based on the authorized users rather than assigned users.
- Ensures that inheritance does not undermine SSD policies.
- Reduce the number of potential permissions that can be made available to a user by placing constraints on the users that can be assigned to a set of roles.

\[ \forall (rs, n) \in SSD, \forall t \subseteq rs \mid |t| \geq n \Rightarrow \bigcap_{r \in t} \text{authorized users}(r) = \emptyset \]
DSD

- Places constraints on the users that can be assigned to a set of roles, thereby reducing the number of potential prms that can be made available to a user.
- Constraints are across or within a user’s session.
- No user may activate \( n \) or more roles from the roles set in each user session.
- *Timely Revocation of Trust* ensures that prms do not persist beyond the time that they are required for performance of duty.

\[
DSD \subseteq (2^{\text{ROLES} \times N}) \\
\forall rs \in 2^{\text{ROLES}}, n \in N, (rs,n) \in DSD \Rightarrow n \geq 2^{|rs|} \geq n \text{ and } \\
\forall s \in \text{SESSIONS}, \forall rs \in 2^{\text{ROLES}}, \forall \text{role\_subset} \subseteq rs, \forall \text{role\_subset} \subseteq \text{session\_role}(s) \Rightarrow |\text{role\_subset}| < n
\]
DSD (cont)

Roles

Cashier

Supervisor

Closes Cashier Role session
Close Cash Drawer
Opens Supv Role session
Open Cash Drawer

Reduce COI
Correct Error
Where we are going....
Current Environment

- Legacy Applications use ACL
- Roles are application specific
- All roles do not follow organizational functions
- Developers and PMs need to think about App roles in their design phase
- Jan 16th Apps will use current mechanism
In Progress

- Developed and Demo Etrust AD and LDAP prototype
- Downloaded and installed NIST Solaris RBAC prototype application
- Coding an XML prototype RBAC database and application
- Exploring CA Identity Manager
- Working on modifying current SEAT process to take advantage of Access Control Groups, then RBAC
- Working on modifying web-based apps to use RBAC₁ implementation
Near Future

- Roles Analysis for new apps
- LPI, K-Reg, and new apps will use RBAC model
- Legacy Apps will continue current AC model
- SEAT will have to support both AC models
- Proposed NIST Standard for RBAC will become a Fed Gov Standard
Future

- All Apps are migrated to a RBAC$_2$
- Roles are centrally managed, but with distributed delegated role assignments and user management
- Expert System module automates most tasks required for central role management
Final Thoughts

- RBAC is a phased approach with increasing level of effort.
- Role engineering is essential in any RBAC rollout.
- RBAC has an up front and steep economic impact, but decreases with time.
- Implementing RBAC requires yet another modification to legacy application.
- SEAT RBAC may not be compatible with TFWeb or any other implementation that uses COTS for their solution. Apps would have to be modified yet again to support this change.
Final Thoughts (cont)

- TPIS is just the user component to a RBAC system.
- A person's cyberID is a set of roles granted him or her access to an object.
- RBAC will free up application owners from the task of account approval.
- Our RBAC still allows for DAC and MAC.
- This RBAC model is applicable for both the Virtual and Physical access policy development.
SEAT is like Sky Diving....
You prepare and get ready for the inevitable..
....the time comes to execute....
...you try to stabilize...
...and hope that everything works at pull time...
...and if all works well, you sail into the sunset.

Knowing that you have your reserve on your back.
QUESTIONS...COMMENTS??