Role-Based Access Control

Overview

- **SSD**
- **ROLE Hierarchy (RH)**
- **User Assignment (UA)**
- **Permission Assignment (PA)**
- **USERS**
- **SESSIONS**
- **ROLES**
- **OPS**
- **OBS**
- **PRMS**

Diagram:

- Arrows indicate relationships and flows between components:
  - **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!!!!
“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 develop cost and time frame.”

-The Economic Impact of Role-Based Access Control

The Time is NOW!
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.

“Ideally, the [RBAC] system is clearly defined and agile, making the addition of new applications, roles, and employees as efficient as possible”
Role-Based AC

Individuals

Roles

Resources

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**

Least Privileged

Most Complex

**Effort**

**RBAC Model**

**RBAC\(_3\)**
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

Diagram:
- USERS
- ROLES
- OPERATIONS (OPS)
- OBS
- SESSIONS
- user_sessions
- session_roles

(UA) User Assignment
(PA) Permission Assignment

PRMS
USERS

Person

Intelligent Agent

Proces
ROLES

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

SQL
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.
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

\[
\text{ROLES set} \quad \begin{array}{c}
\text{User.F1} \\
\text{User.F2} \\
\text{User.F3} \\
\text{User.DB1}
\end{array}
\]

\[
\text{USERS set} \quad \begin{array}{c}
\text{User.DB1} \\
\text{User.DB1}
\end{array}
\]

\[
assigned\_user : (r : \text{ROLES}) \rightarrow 2^{\text{users}}
\]

\[
assigned\_user(r) = \{ u \in \text{USERS} \mid (u, r) \in UA \}
\]
PRMS (permissions)

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

User.DB1
- View
- Update
- Append

User.F1
- Read
- Write
- Execute

\[ PRMS = 2^{(OPS \times OBS)} \]
A prms can be assigned to one or more roles

A role can be assigned to one or more prms

\( PA \subseteq PRMS \times ROLES \)
PA (prms assignment)

Mapping of role $r$ onto a set of permissions

$\text{ROLES set}$

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

$\text{PRMS set}$

- Read
- Write
- Execute
- View
- Update
- Append
- Create
- Drop

$\text{SQL}$

\[
\text{assigned \_ permissions}(r : \text{ROLES}) \rightarrow 2^{\text{PRMS}}
\]

\[
\text{assigned \_ permissions}(r) = \{ p \in \text{PRMS} \mid (p, r) \in PA \}
\]
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

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

Mapping of permissions to objects

PRMS set
- Open
- Close
- View
- Update
- Append
- Create
- Drop

Objects
- BLD1.door2
- DB1.table1
- SQL

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

\[
\text{SESSION} \rightarrow 2^{\text{ROLES}}
\]

\[
\text{session\_roles}(s : \text{SESSIONS}) \subseteq \{ r \in \text{ROLES} \mid (\text{session\_users}(s), r) \in \text{UA} \}
\]

DB1.table1.session
SESSIONS

Permissions available to a user in a session.

DB1.ADMIN

- View
- Update
- Append
- Create
- Drop

DB1.table1.session

\[
\begin{align*}
\text{avail}_{session\_persm}(s:SECTIONS) & \rightarrow 2^{PRMS} \\
\bigcup_{r:session\_rules(s)} \text{permissions}(r)
\end{align*}
\]
Hierarchal RBAC
Lattice Hierarchy

Director

Project Lead 1
- Production Engineer 1
- Engineer 1

Project Lead 2
- Quality Engineer 1
- Engineer 1

- Production Engineer 2
- Engineer 2
- Quality 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 \( r_2 \)

  \[
  \begin{array}{ll}
  \text{User} & \text{Guest} \\
  \text{r-w-h} & \text{-r-}
  \end{array}
  \]
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 \( r_2 \)

\( r_1 \supseteq r_2 \Rightarrow authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1) \)
\( ^\wedge authorized\_users(r_1) \subseteq authorized\_users(r_2) \)
authorized users

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

ROLES set

Admin.DB1
User.DB2
User.DB3
User.DB1

permissions object

User.DB1

First Tier USERS set

User.DB1

\[ \text{authorized } \_ \_ \text{users}(r) = \{ u \in \text{USERS} | r \geq r(u, r) \in UA \} \]
authorized permissions

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

ROLES set

User.DB1
User.DB2
User.DB3
Admin.DB1

PRMS set

• View
• Update
• Append
• Create
• Drop

SQL

authorized_permissions(r : ROLES) → 2^{PRMS}
authorized_permissions(r) = \{ p ∈ PRMS | r' ≥ r, (p,r') ∈ PA \}
Limited RH

A restriction on the immediate descendants of the general role hierarchy

Role2 inherits from Role1

- Role3

Role3 does not inherit from Role1 or Role2

∀r, r_1, r_2 ∈ ROLES, r ≥ r_1 ∧ r ≥ r_2 ⇒ r_1 = r_2
Limited RH (cont)

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.

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

\[
\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{rs} \text{assigned_users}(r) = \emptyset
\]
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.

∀(rs, n) ∈ SSD, ∀t ⊆ rs : t ≥ n ⇒ \( \bigcap_{ref} \text{authorized \_users}(r) = \emptyset \)
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.

\[
DSD \subseteq \left(2^{\text{ROLES}}\times N\right) \\
\forall rs \in 2^{\text{ROLES}}, n \in N, (rs, n) \in DSD \Rightarrow n \geq 2^{|rs|}; n \geq n, and \\
\forall s \in \text{SESSIONS}, \forall rs \in 2^{\text{ROLES}}, \forall \text{role subset} \in 2^{\text{ROLES}}, \forall n \in N, (rs, n) \in DSD, \text{role subset} \subseteq rs, \text{role subset} \subseteq \text{session_role(s)} \Rightarrow \text{role subset} < n
\]
DSD (cont)

Roles

Cashier

Supervisor

Cashier inherits Supervisor

Closes Cashier Role session

Close Cash Drawer

Opens Supv Role session

Open Cash Drawer

Reduce COI

Correct Error
QUESTIONS...COMMENTS??