Lecture 5: DAC in RBAC, Role Hierarchies

INFS 767
Secure Electronic Commerce
Fall 1999

Lecture 5
DAC in RBAC
Role Hierarchies

Prof. Ravi Sandhu

RBAC96

ROLE HIERARCHIES

USER-ROLE ASSIGNMENT

PERMISSION-ROLE ASSIGNMENT

USERS

ROLES

PERMISSIONS

SESSIONS

CONSTRAINTS

© Ravi Sandhu 1998
DAC in RBAC

- RBAC is policy neutral and highly expressive
- Can RBAC enforce MAC and DAC policies?
- Simulation of MAC in RBAC is demonstrated earlier.
- Variety of DAC policies are simulated.

Owner Centric DAC

- The creator of an object becomes its owner.
- There is only one owner of an object
  - Ownership remain fixed.
  - It can be transferred to another user.
- Destruction of object can only be done by its owner.
Variations of DAC

- Strict DAC
- Liberal DAC

Strict DAC

- Ownership cannot be transferred.
- Only owner has a discretionary authority to grant access to an object.
- Example:
  - Alice has created an object (he is owner) and grants access to Bob. Now Bob cannot grant propagate the access to another user.
Lecture 5: DAC in RBAC, Role Hierarchies

Liberal DAC

- Owner can delegate discretionary authority for granting access to other users.
  - One Level grant
  - Two Level Grant
  - Multilevel Grant

One Level Grant

- Owner can delegate authority to another user but they cannot further delegate this power.

Alice  <->  Bob  <->  Charles
Two Level Grant

- In addition to a one level grant, the owner can allow some users to delegate grant authority to other users.

```
Alice -> Bob -> Charles -> Dorothy
```

Multilevel DAC

- In this case, the power to delegate the power to grant implies that this authority can itself be delegated.

```
Alice -> Bob -> Charles -> Dorothy
```

© Ravi Sandhu 1998
DAC with change of Ownership

- This variation allows a user to transfer ownership of an object to another user.
- Can be combined with strict or liberal DAC with all variations.

Revocation

- Grant-Independent Revocation.
- Grant-Dependent Revocation.
Common Aspects

- Creation of an object in the system requires the simultaneous creation of
  - three administrative roles
    - OWN_O, PARENT_O, PARENTwithGRANT_O
  - One regular role
    - READ_O

Administration of roles associated with object O

Administrative role hierarchy
Common Aspects II

- We require simultaneous creation of Eight Permissions
  - canRead_O
  - destroyObject_O
  - addReadUser_O, deleteReadUser_O
  - addParent_O, deleteParent_O
  - addParentWithGrant_O, deleteParentWithGrant_O

Roles and associated Permissions

- OWN_O
  - destroyObject_O, addParentWithGrant_O, deleteParentWithGrant_O
- PARENTwithGRANT_O
  - addParent_O, deleteParent_O
- PARENT_O
  - addReadUser_O, deleteReadUser_O
- READ_O
  - canRead_O
Common Aspects III

- Destroying an object O requires deletion of four roles and eight permissions in addition of destroying the object O.

Strict DAC in RBAC96

- Cardinality constraints as:
  - Role OWN_O = 1
  - Role PARENTwithGARNT_O = 0
  - Role PARENT_O = 0
One level DAC in RBAC96

- Cardinality constraints as:
  - Role OWN_O = 1
  - Role PARENTwithGARNT_O = 0

Two Level DAC in RBAC96

- Cardinality constraints as:
  - Role OWN_O = 1
Multilevel DAC in RBAC96

- At the time of creation of an object the association of permissions to roles are redefined as:
  - OWN_O
    - destroyObject_O
  - PARENTwithGRANT_O
    - addParent_O, deleteParent_O,
      addParentWithGrant_O, deleteParentWithgrant_O
  - PARENT_O and READ_O are same
  - Cardinality constraints as of Two level DAC.

Multiple Ownership

- Can be accommodated by allowing users to be added to OWN_O
- All members of OWN_O has identical powers, including the ability to revoke other owners
- grant-independent revoke
Grant-Dependent Revoke

- Only granter can revoke access
- When owner authorizes user U, the roles U_PARENT_O and U_READ_O and permissions addU_ReadUser_O and deleteU_ReadUser_O are automatically created.
- New permissions are assigned to U_PARENT_O role.

Grant-Dependent Revoke cont..

- canRead_O permission is assigned to U_READ_O role at the time of creation.
- U_PARENT_O manages the membership assignments to U_READ_O.
- Cardinality constraint of U_PARENT_O is 1 and its membership cannot be changed.
- Owner can revoke every thing, thus OWN_O is senior to all U_PARENT_O.
Grant-Dependent Revoke cont..

- Similar revocation can be simulated with respect to PARENT_O and PARENTwithGRANT_O roles.

```
U1_PARENT_O ------ U1_READ_O

U2_PARENT_O ------ U2_READ_O

Un_PARENT_O ------ Un_READ_O

READ_O role associated with members of PARENT_O
```
DAC in RBAC

- Many other variations can be simulated in the similar way.
- Informal constructions can be formalized using RBAC96.
- Objects can be grouped together to overcome the high number of roles.
- Results confirm that RBAC is policy neutral and can accommodate DAC and MAC.

ROLE HIERARCHIES

- Inheritance hierarchies
  - permission inheritance
  - user inheritance
- Activation hierarchies
  - role membership versus role activation
EXAMPLE ROLE HIERARCHY

INTERPRETATIONS

Director (DIR)

Project Lead 1 (PL1)
  Production 1 (P1)
    Engineer 1 (E1)

Project Lead 2 (PL2)
  Production 2 (P2)
    Engineer 2 (E2)

Quality 1 (Q1)

Quality 2 (Q2)

Engineering Department (ED)

Employee (E)

PROJECT 1

PROJECT 2

ALTERNATIVES

- separate inheritance and activation hierarchies
  - this paper
- single inheritance and activation hierarchy
  - most common approach, including RBAC96
- activation hierarchy only, no inheritance
  - alternative identified in NIST RBAC model
- inheritance hierarchy only, no activation hierarchy
  - does not seem to be useful
Lecture 5: DAC in RBAC, Role Hierarchies

**LBAC: LIBERAL *-PROPERTY**

```
                H
               / \
              M1 M2
             /   \
            L    
```

Read      Write
-      +

**LBAC: LIBERAL *-PROPERTY**

**DUAL ROLE SIMULATION**

```
                HR
               /  \
              M1R M2R
             /    \
            LR    
```

```
                LW
               /  \
              M1W M2W
             /    \
            HW    
```

Read      Write
-      +
Lecture 5: DAC in RBAC, Role Hierarchies

LBAC: STRICT *-PROPERTY

<table>
<thead>
<tr>
<th>H</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>M2</td>
</tr>
<tr>
<td>L</td>
<td>-</td>
</tr>
</tbody>
</table>

Read  Write

LBAC: STRICT *-PROPERTY
DUAL ROLE SIMULATION

<table>
<thead>
<tr>
<th>HR</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1R</td>
</tr>
<tr>
<td>LR</td>
</tr>
</tbody>
</table>
LBAC: STRICT *-PROPERTY
SIMULATION BY PRIVATE ROLES

HR

M1R         M2R

LR

© Ravi Sandhu 1998

LBAC: STRICT *-PROPERTY
SIMULATION BY PRIVATE ROLES

HW

HR

M1W         M2W

M1R         M2R

LW

LR

© Ravi Sandhu 1998
**LBAC: STRICT *-PROPERTY SIMULATION BY PRIVATE ROLES**

![Diagram](image)

**DYNAMIC SEPARATION OF DUTIES**

- Roles in dynamic SOD
  - cannot have common seniors in role inheritance hierarchy, but
  - can have common seniors in role activation hierarchy
EXAMPLE ROLE HIERARCHY

INTERPRETATIONS

Director (DIR)

Project Lead 1 (PL1)
- Production 1 (P1)
- Engineer 1 (E1)

Project Lead 2 (PL2)
- Production 2 (P2)
- Engineer 2 (E2)

Engineering Department (ED)

Employee (E)

PROJECT 1

PROJECT 2

ACTIVATION HIERARCHIES

A

B

C

D

E

A

B

C

D

E
CONCLUSION

- separate inheritance and activation hierarchies
  - this paper
- single inheritance and activation hierarchy
  - most common approach, including RBAC96
- activation hierarchy only, no inheritance
  - alternative identified in NIST RBAC model
- inheritance hierarchy only, no activation hierarchy
  - does not seem to be useful