Role-Based Access Control (RBAC)

Prof. Ravi Sandhu
Executive Director and Endowed Chair

Lecture 4

ravi.utsa@gmail.com
www.profsandhu.com
Access Control

- Discretionary Access Control (DAC), 1970
- Mandatory Access Control (MAC), 1970
- Role Based Access Control (RBAC), 1995
- Attribute Based Access Control (ABAC), ???

Fixed policy

Flexible policy
Access Control

Discretionary Access Control (DAC), 1970

Ownership gives discretion

Policy neutral

Role Based Access Control (RBAC), 1995

Mandatory Access Control (MAC), 1970

One-directional information flow

Attribute Based Access Control (ABAC), ????

Fixed policy

Flexible policy
The RBAC Story

World-Leading Research with Real-World Impact!

RBAC: Role-Based Access Control

- Access is determined by roles
- A user’s roles are assigned by security administrators
- A role’s permissions are assigned by security administrators

Is RBAC MAC or DAC or neither?

- RBAC can be configured to do MAC
- RBAC can be configured to do DAC
- RBAC is policy neutral

RBAC is neither MAC nor DAC!
RBAC96 Model
RBAC96 Model Family

ROLE HIERARCHIES

USER-ROLE ASSIGNMENT

SESSIONS

PERMISSIONS-ROLE ASSIGNMENT

PERMISSIONS

© Ravi Sandhu

World-Leading Research with Real-World Impact!
RBAC96 Model Family

RBAC3
ROLE HIERARCHIES + CONSTRAINTS

RBAC1
ROLE HIERARCHIES

RBAC0
BASIC RBAC

RBAC2
CONSTRAINTS
Founding Principles of RBAC96

- **Abstraction** of Privileges
  - Credit is different from Debit even though both require read and write

- **Separation** of Administrative Functions
  - Separation of user-role assignment from role-permission assignment

- **Least Privilege**
  - Right-size the roles
  - Don’t activate all roles all the time
  - Limit roles of a user
  - Limit users in a role

- **Separation of Duty**
  - Static separation: purchasing manager versus accounts payable manager
  - Dynamic separation: cash-register clerk versus cash-register manager
A role brings together
  - a collection of users and
  - a collection of permissions

These collections will vary over time
  - A role has significance and meaning beyond the particular users and permissions brought together at any moment
Groups are often defined as
- a collection of users

A role is
- a collection of users and
- a collection of permissions

Some authors define role as
- a collection of permissions

Most Operating Systems support groups
- BUT do not support selective activation of groups

Selective activation conflicts with negative groups (or roles)
HIERARCHICAL ROLES

Physician

Primary-Care Physician

Specialist Physician

Health-Care Provider
HIERARCHICAL ROLES

Supervising Engineer

Hardware Engineer

Software Engineer

Engineer
EXAMPLE ROLE HIERARCHY

Director (DIR)

- Project Lead 1 (PL1)
  - Production 1 (P1)
  - Quality 1 (Q1)
  - Engineer 1 (E1)

- Engineering Department (ED)

- Project Lead 2 (PL2)
  - Production 2 (P2)
  - Quality 2 (Q2)
  - Engineer 2 (E2)

PROJECT 1

PROJECT 2
EXAMPLE ROLE HIERARCHY

PROJECT 1

Director (DIR)

Project Lead 1 (PL1)

Production 1 (P1)

Quality 1 (Q1)

Engineer 1 (E1)

PROJECT 2

Project Lead 2 (PL2)

Production 2 (P2)

Quality 2 (Q2)

Engineer 2 (E2)
EXAMPLE ROLE HIERARCHY

PROJECT 1

Project Lead 1 (PL1)

Project Lead 2 (PL2)

Production 1 (P1)

Production 2 (P2)

Quality 1 (Q1)

Quality 2 (Q2)

Engineer 1 (E1)

Engineer 2 (E2)

PROJECT 2

World-Leading Research with Real-World Impact!
CONSTRAINTS

- **Mutually Exclusive Roles**
  - Static Exclusion: The same individual can never hold both roles
  - Dynamic Exclusion: The same individual can never hold both roles in the same context

- **Mutually Exclusive Permissions**
  - Static Exclusion: The same role should never be assigned both permissions
  - Dynamic Exclusion: The same role can never hold both permissions in the same context

- **Cardinality Constraints on User-Role Assignment**
  - At most $k$ users can belong to the role
  - At least $k$ users must belong to the role
  - Exactly $k$ users must belong to the role

- **Cardinality Constraints on Permissions-Role Assignment**
  - At most $k$ roles can get the permission
  - At least $k$ roles must get the permission
  - Exactly $k$ roles must get the permission
• Formalized in RCL2000 paper
NIST RBAC Model
NIST MODEL: CORE RBAC

USER SESSIONS

USER ASSIGNMENT

ROLE ASSIGNMENT

SESSION ROLES

USER SESSIONS

(UA) User Assignment

(PA) Permission Assignment

SESSIONS

OPS

OBS

PRMS
NIST MODEL: HIERARCHICAL RBAC

(RH) Role Hierarchy

(UA) User Assignment

(PA) Permission Assignment

ROLE

user_sessions

session_roles

OPS

OBS

PRMS

World-Leading Research with Real-World Impact!
SSD IN HIERARCHICAL RBAC

- SSD
- Role Hierarchy
- User Assignment (UA)
- Session Sessions
- Session Roles
- Permission Assignment (PA)
- Users
- Roles
- Sessions

World-Leading Research with Real-World Impact!
DSD IN HIERARCHICAL RBAC

- **(UA) User Assignment**
- **(PA) Permission Assignment**

**DSD**

**USERs**

**ROLEs**

**SESSIONs**

- **user_sessions**
- **session_roles**

**OPS**

**OBS**

**PRMS**

World-Leading Research with Real-World Impact!

© Ravi Sandhu
NIST MODEL FAMILY

Select Core RBAC
Option: Advanced Review

Core RBAC

Hier. RBAC
a. Limited
b. General

Choose a or b
Option: Advanced Review

SSD Relations
a. w/hierarchies
b. wo/hierarchies

Adhere to dependency

DSD Relations

Requirements Package
Compare RBAC96 Model Family

RBAC3
ROLE HIERARCHIES +
CONSTRAINTS

RBAC1
ROLE
HIERARCHIES

RBAC0
BASIC RBAC

RBAC2
CONSTRAINTS
RBAC Administration
• Separation of regular roles and administrative roles

• Formalized in ARBAC97 paper
EXAMPLE REGULAR ROLE HIERARCHY

Director (DIR)

- Project Lead 1 (PL1)
  - Production 1 (P1)
  - Quality 1 (Q1)
  - Engineer 1 (E1)

- Engineering Department (ED)

- Employee (E)

- PROJECT 1

- Project Lead 2 (PL2)
  - Production 2 (P2)
  - Quality 2 (Q2)
  - Engineer 2 (E2)

- PROJECT 2
EXAMPLE ADMIN ROLE HIERARCHY

Senior Security Officer (SSO)

Department Security Officer (DSO)

Project Security Officer 1 (PSO1)

Project Security Officer 2 (PSO2)
MAC in RBAC
MAC Liberal ★-Property

World-Leading Research with Real-World Impact!
RBAC96 Liberal ★-Property

M1R \( \leftrightarrow \) M2R  
\( \rightarrow \) Read  
\( \leftarrow \) Write

\( \rightarrow \) M1W \( \leftrightarrow \) M2W

\( \rightarrow \) LW  
\( \leftarrow \) HW

World-Leading Research with Real-World Impact!
• user $\in xR$, user has clearance $x$
  user $\in LW$, independent of clearance

• Constraints
  – session $\in xR$ iff session $\in xW$
  – read can be assigned only to $xR$ roles
  – write can be assigned only to $xW$ roles
  – $(O,\text{read})$ assigned to $xR$ iff
    $(O,\text{write})$ assigned to $xW$
• user $\in xR$, user has clearance $x$
  user $\in LW$, independent of clearance

• Constraints
  - session $\in xR$ iff session $\in xW$
  - read can be assigned only to xR roles
  - write can be assigned only to xW roles
  - $(O,\text{read})$ assigned to xR iff
    $(O,\text{write})$ assigned to xW

NIST Model cannot express these constraints
MAC Strict ★-Property

Symbol: M1, H, L, M2

Diagram:
- Quadrilateral with M1 and M2 at opposite corners.
- H and L at adjacent corners.

Notation:
- Read
- + and - signs indicate relationships.
RBAC96 Strict ★-Property

HR
M1R

M2R
LR

M1W LW HW M2W

Read
Write

World-Leading Research with Real-World Impact!
- user ∈ xR, user has clearance x
  user ∈ \{LW,HW,M1W,M2W\}, independent of clearance

- Constraints
  - session ∈ xR iff session ∈ xW
  - read can be assigned only to xR roles
  - write can be assigned only to xW roles
  - (O,read) assigned to xR iff (O,write) assigned to xW
DAC in RBAC
Variations of Grant

- **Strict DAC**
  - Only owner has discretionary authority to grant access to an object.
  - Example:
    - Alice has created an object (she is owner) and grants access to Bob. Now Bob cannot grant propagate the access to another user.

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

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

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

- **Grant-Independent Revocation**
  - Any authorized revoker can revoke
  - Easier to do in RBAC

- **Grant-Dependent Revocation**
  - Only original grantor can revoke
  - Need additional roles to accomplish in RBAC
Common Aspects

- Creation of an object $O$ in the system requires the simultaneous creation of
  - 3 administrative roles
    - OWN\(_O\), PARENT\(_O\), PARENTwithGRANT\(_O\)
  - 1 regular role
    - READ\(_O\)
- Also simultaneous creation of 8 Permissions
  - canRead\(_O\)
  - destroyObject\(_O\)
  - addReadUser\(_O\), deleteReadUser\(_O\)
  - addParent\(_O\), deleteParent\(_O\)
  - addParentWithGrant\(_O\), deleteParentWithGrant\(_O\)
- Destroying an object $O$ requires deletion of 4 roles and 8 permissions in addition of destroying the object $O$
Common Aspects

Administration of roles associated with object O

OWN_O ➔ PARENTwithGRANT_O ➔ PARENT_O ➔ READ_O

Role permissions

- destroyObject_O
- addParentWithGrant_O
- deleteParentWithGrant_O
- addParent_O
- deleteParent_O
- addReadUser_O
- deleteReadUser_O
- canRead_O

World-Leading Research with Real-World Impact!
Grant Variations in RBAC96

- **Strict DAC cardinality constraints**
  - Role OWN_O = 1
  - Role PARENTwithGRANT_O = 0
  - Role PARENT_O = 0

- **One-level grant cardinality constraints**
  - Role OWN_O = 1
  - Role PARENTwithGRANT_O = 0

- **Two-level grant cardinality constraints**
  - Role OWN_O = 1
Grant Variations in RBAC96

- **Strict DAC cardinality constraints**
  - Role OWN_O = 1
  - Role PARENTwithGRANT_O = 0
  - Role PARENT_O = 0

- **One-level grant cardinality constraints**
  - Role OWN_O = 1
  - Role PARENTwithGRANT_O = 0

- **Two-level grant cardinality constraints**
  - Role OWN_O = 1

NIST Model cannot express these constraints
Grant-Dependent Revoke in RBAC96

READ_O role associated with members of PARENT_O
OM-AM and PEI
<table>
<thead>
<tr>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectives</td>
<td>Model</td>
</tr>
<tr>
<td>Architecture</td>
<td>Mechanism</td>
</tr>
</tbody>
</table>

**OM-AM**

*World-Leading Research with Real-World Impact!*

© Ravi Sandhu
PEI Models

Vertical View
Looks Across Layers

Horizontal view
Looks at Individual layer

Security and system goals (objectives/policy)

Policy models

Enforcement models

Implementation models

Trusted Computing Technology (mechanisms/implementation)

Necessarily Informal

Formal/quasi-formal

System block diagrams, Protocol flows

Pseudo-code

Actual Code

Idealized

Enforceable (Approximate)

Codeable
RBAC: SERVER PULL

Client → User-role Authorization Server → Server

E model

World-Leading Research with Real-World Impact!
RBAC: CLIENT PULL

E model

Client

User-role Authorization
Server

Server
RBAC: PROXY-BASED

E model

Client — Proxy Server — Server

User-role Authorization Server
MAC or LBAC or BLP (or Biba)

- BLP enforces one-directional information flow in a lattice of security labels
- BLP can enforce one-directional information flow policies for:
  - Confidentiality
  - Integrity
  - Separation of duty
  - Combinations thereof
MAC

Security and system goals (objectives/policy)

Policy models

Enforcement models

Implementation models

Trusted Computing Technology (mechanisms/implementation)

Confidentiality

Integrity

Separation

One-Direction

Information Flow
DAC

Owner Discretion
Access Matrix
Capabilities, Access Control Lists, Access Control Relations

Security and system goals (objectives/policy)

Policy models

Enforcement models

Implementation models

Trusted Computing Technology (mechanisms/implementation)

Necessarily Informal
Formal/quasi-formal
System block diagrams, Protocol flows
Pseudo-code
Actual Code

Vertical View
Looks Across Layers

Horizontal view