OM-AM Framework

- Objectives (policies, requirements, etc)
- Models
- Architectures
- Enforcement Mechanisms
- Assurance

Access Control Architecture

- An abstract overview
Access Control Architecture

- REQUESTING ROLE
- ACCESS CONTROL ENFORCEMENT
- RESOURCE
- AUTHENTICATION
- ACCESS CONTROL DECISION
- ACCESS CONTROL POLICY RETRIEVAL
- POLICY MANAGEMENT

- RETRIEVAL OF ACCESS RIGHTS, PRIVILEGES, & ATTRIBUTES
- INFORMATION LABELING MANAGEMENT
- AUTHORIZATION & PRIVILEGE MANAGEMENT

Access Control Models

- Evolvement of AC Models
  - Identity-based
    - AC Matrix, DAC, etc
  - Label-based
    - MAC
  - Function/duty/task/role-based:
    - RBAC, etc
  - Attribute-based:
    - UCON
    - DRM
    - Trusted Management

-- Thanks to John McLaughlin to share this slide
Role-Based Access Control (RBAC)

ISA 767, Secure Electronic Commerce
Xinwen Zhang, xzhang6@gmu.edu
George Mason University
Fall 2005

### RBAC in OM-AM

<table>
<thead>
<tr>
<th>What?</th>
<th>Objective</th>
<th>Policy Neutral</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model</td>
<td>RBAC96 Model, ARBAC97, etc.</td>
</tr>
<tr>
<td></td>
<td>Architecture</td>
<td>Sever-pull, User-pull, etc.</td>
</tr>
<tr>
<td></td>
<td>Mechanism</td>
<td>Secure Cookies, Digital Certificates, SAML/XACML, etc.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How?</th>
<th>OM-AM Framework</th>
<th>RBAC System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Assurance</td>
<td></td>
</tr>
</tbody>
</table>
Outline

- RBAC96 Model
- Using RBAC96 to configure Policies
  - SOD, least of privilege, etc
- Distributed RBAC Architecture
  - User-pull, server-pull
- RBAC implementation mechanisms
  - Role certificate

RBAC96 Model

- Role-based access control models
- The most cited paper in access control
  - Cited by 938 by google scholar
Role-Based Access Control

- A user’s permissions are determined by the user’s roles not identity or clearance
- RBAC provides a framework to help in articulating policy
  - By utilizing users, roles, permissions, and their relations.
- The main goal of RBAC is to facilitate security management
- Range from simple to sophisticated
  - Users from hundreds to thousands
  - Roles from tens to hundreds
  - Role hierarchies, constraints,

RBAC Security Principles

- Least Privilege
- Separation of Duty (SoD)
- Separation of administration and access
- Abstractions
  - Abstract operations
    - Open account, close account
  - Abstraction object
RBAC96 Model

- Policy neutral
  - Can be configured to do MAC
    - Roles simulate clearances
  - Can be configured to do DAC
    - Roles simulate identity
- Base model for NIST’s RBAC standards

RBAC96 Family of Models

- RBAC\(_3\)
  - (Role Hierarchies + Constraints)
- RBAC\(_1\)
  - (Role Hierarchies)
- RBAC\(_2\)
  - (Constraints)
- RBAC\(_0\)
  - (Base Model)
Users

- In RBAC, a user is a human being though can be generalized to include other active agents
- Each individual should be known as exactly one user
Permissions

- An approval of a **particular mode of access to one or more objects** in the system
  - A collection of objects and rights
- Primitive permissions
  - read, write, append, execute, etc.
- Abstract permissions
  - credit, debit, inquiry, create/delete account, etc.
- System permissions
  - **auditorObject permissions**

Permissions (cont)

- Always positive. No Negative permission.
  - An RBAC system is closed system
  - Denial of access is modeled as a constraint
- A permission for a single object and a set of objects
  - Depends on applications
- Abstract Objects
  - Data objects, Resource objects, Service objects, ...
- No duties or obligations
  - outside scope of access control
Roles

- Role is a job function or job title within the organization
- 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

Role vs. Group

- Role
  - a collection of users and
  - a collection of permissions
  - Dynamic permissions within different sessions
  - Role membership and permissions are easy to determine - controlled by centralized in a few users
- Group
  - a collections of users-easy to determine
  - Permissions of a group are difficult to determine
    - Unix: /etc/passwd and /etc/group
    - Easy to determine a user’s group membership
    - Since each user can define the permissions to his own files, it’s difficult to enumerate.
User-Role Assignment (UA)

- A user can be a member of many roles
- Each role can have many users as members

Sessions

- A session is a mapping of one user to possibly many roles
- Each session is belong to only one user
- A user can invoke (activate) multiple sessions (one to many relationship)
- In each session a user can invoke (activate) any subset of roles that the user is a member of
Permission-Role Assignment (PA)

- A permission can be assigned to many roles
- Each role can have many permissions

RBAC₀ Definition

- $U$, $R$, $P$, and $S$
  - Users, roles, permissions, sessions
- $PA \subseteq P \times R$: permission-role assignment
- $UA \subseteq U \times R$: user-role assignment
- $user: S \rightarrow U$,
- $roles: S \rightarrow 2^R$,  
  - $roles(s) \subseteq \{r \mid (user(s), r) \in UA\}$,
- $Permissions: S \rightarrow 2^P$
  - $Permissions(s) \subseteq U_{r \in roles(s)}\{p \mid (p, r) \in PA\}$
**Why RBAC**

- Fewer relationships to manage
  - from $O(mn)$ to $O(m+n)$, where $m$ is the number of users and $n$ is the number of permissions
- Roles add a useful level of indirection (middle layer)

---

**RBAC$_1$: RBAC$_0$ + Role Hierarchy**

![Diagram showing RBAC hierarchy with user-role and permission-role assignments](image_url)

- Users
- Roles
- Permissions
- User-Role Assignment (UA)
- Permission-Role Assignment (PA)
- Role Hierarchy (RH)
- Sessions
Role Hierarchy: Transitive inheritance of permissions

Primary-Care Physician

Physician

Specialist Physician

Health-Care Provider

Role Hierarchy: Multiple inheritance of permissions

Supervising Engineer

Hardware Engineer

Software Engineer

Engineer
Role Hierarchy: Private Role

Hardware Engineer’ → Hardware Engineer → Engineer

Supervising Engineer

Software Engineer’ → Software Engineer

Role Hierarchy Example

Director (DIR)

Project Lead 1 (PL1)
Production Engineer 1 (PE1)
Engineer 1 (E1)

Quality Engineer 1 (QE1)

Project Lead 2 (PL2)
Production Engineer 2 (PE2)
Engineer 2 (E2)

Engineering Department (ED)

Employee (E)
Role Hierarchy Example

Semantics of RH

- User inheritance
  - $r_1 \geq r_2$ means every user that is a member of $r_1$ is also a member of $r_2$
- Permission inheritance
  - $1 \geq r_2$ means every permission that is authorized for $r_2$ is also authorized for $r_1$
- Activation inheritance
  - $r_1 \geq r_2$ means that activating $r_1$ will also activate $r_2$
**RBAC\textsubscript{1} Definition**

- \( U, R, P, S, PA, UA \) and \textit{user} are same as RBAC\textsubscript{0},
- \( RH \subseteq R \times R \), a partial order with dominance relation \( \geq \),
- \textit{roles}: \( S \rightarrow 2^R \),
  - \( \text{roles}(s) \subseteq \{ r \mid (\exists r' \geq r)[(\text{user}(s), r') \in UA] \} \),
- \textit{Permissions}: \( S \rightarrow 2^P \)
  - \( \text{Permissions}(s) \subseteq \bigcup_{r \in \text{roles}(s)} \{ p \mid (\exists r'' \leq r)[(p, r'') \in PA] \} \)

**RBAC\textsubscript{2}: RBAC\textsubscript{0} + Constraints**

![Diagram showing User-Role Assignment (UA), Permission-Role Assignment (PA), Users, Roles, Permissions, Sessions, and Constraints]
RBAC\textsubscript{2} Definition

- RBAC\textsubscript{2} is unchanged from RBAC\textsubscript{0} except for requiring that there be a collection of constraints that determine whether or not values of various components of RBAC\textsubscript{0} are acceptable. Only acceptable values will be permitted.

Constraints

- Mutually Exclusive Roles (Separation of Duty Constraints)
  - Static Exclusion (static SoD): The same individual user can never hold mutually exclusive roles (by UA)
  - Dynamic Exclusion (dynamic SoD): The same individual user can never hold mutually exclusive roles in single session
Constraints

- **Mutually Exclusive Roles (Separation of Duty Constraints)**
  - **Static Exclusion (static SoD):** Two mutually exclusive roles cannot be assigned with the same permissions.
  - **Dynamic Exclusion (dynamic SoD):** Two mutually exclusive roles can be assigned with the same permissions but cannot be activated at the same time by different users.

Constraints

- **Mutually Exclusive Permissions (Separation of Duty Constraints)**
  - **Static Exclusion (static SoD):** The same role should never be assigned to mutually exclusive permissions.
  - **Dynamic Exclusion (dynamic SoD):** The same role can never hold mutually exclusive permissions in single session.
Project Topics

- Are mutually exclusive roles, and permissions equivalent?
  - How to define equivalence
  - Can one of them can simulate the other two?

- How to verify SoD objectives has been achieved by constraints?
  - The mapping between model and policies
  - Sensitive tasks and the permissions they require need to be identified
  - Define algorithms, complexity, and implementation tradeoffs

Project Topics

- Conflicts between exclusive roles/permissions
  - When may conflicts happen?
  - How to find conflicts in a pre-defined constraints?

- Implementation issues
  - Event mechanisms for role activation
**Constraints**

- **Cardinality Constraints on User-Role Assignment**
  - At most/at least/exactly \( k \) users can/must belong to the role

- **Cardinality Constraints on Permission-Role Assignment**
  - At most/at least/exactly \( k \) roles can/must get the permission

- **On activation**
  - at most/at least/exactly \( k \) users can activate a role

---

**Constraint**

- **Prerequisite roles**
  - A user can be assigned to role \( A \) only if he already is assigned to role \( B \).
    - Generally role \( A \geq B \)
    - Is an incomparable prerequisite role meaningful?
  - A permission \( p \) can be assigned to a role only if the role already possesses permission \( q \).
  - More like an administrative issue.
**RBAC\(_3\): RBAC\(_0\) + RH + Constraints**

- **RBAC\(_0\)**: Base RBAC model.
- **RH** (Role Hierarchy): Hierarchical organization of roles.
- **Constraints**: Rules that govern the system's behavior.
- **User-Role Assignment (UA)**:
  - Users connect to roles.
- **Permission-Role Assignment (PA)**:
  - Roles connect to permissions.
- **Sessions**: Temporary connections between users and sessions.

**OM-AM Framework**

- **What?**: Objectives (policies, requirements, etc).
- **How?**:
  - Models
  - Architectures
  - Enforcement Mechanisms
  - Assurance
Policy

- Role-based authorization constraint language (RCL2000)
  - Least privilege, SSoD, DSod, etc.

RABC Mechanisms

- Secure Cookies
- Smart Certificates
- X.509 Certificates
- Implementing Secure Attribute Services on the web using above mechanism with server pull and user pull architectures
Attribute Certificates


X.509 Certificate

- Digitally signed by a certificate authority
  - to confirm the information in the certificate belongs to the holder of the corresponding private key
- Contents
  - version, serial number, subject, validity period, issuer, optional fields (v2)
  - subject’s public key and algorithm info.
  - extension fields (v3)
  - digital signature of CA
- Binding users to keys
- Certificate Revocation List (CRL)
X.509 Certificate

Certificate Content:

Certificate:

Data:

Version: v3 (0x2)
Serial Number: 5 (0x5)
Signature Algorithm: SHA1 With RSA Encryption
Issuer: CN=Data.Ist.psu.edu, O=CMU, C=US
Value:

Not Before: Tue Feb 09 03:19:38 1999
Not After: Wed Feb 09 03:19:38 2000
Subject: CN=Data.Ist.psu.edu, O=CMU, C=US

Subject Public Key Info:

Algorithm: SHA1 With RSA Encryption

Public Key:

491bf7e4050450a561be69475d434eb3908

Extension:

Identifier: Certificate Type

Critical: 0

Identifier: Authority Key Identifier

Critical: 0

Key Identifier: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00

Signature:

Algorithm: SHA1 With RSA Encryption

Signature:

Attribute Certificate

- Incorporated into both the ANSI X9.57 standard and X.509.
- An attribute certificate binds attribute information to the certificate’s subject.
- The certificate is digitally signed and issued by an attribute authority.
- An attribute certificate does not contain a public key.
- Therefore, an attribute certificate needs to be used in conjunction with authentication services, such as another certificate (X.509) and SSL.
Bind ID and Attributes

An attribute certificate is linked to an identity certificate by

- the subject name (X.500 name)
- The serial number of the X.509 certificate
- Public key
- CA’s signature
- Other info in X.509 certificate (e.g., email)

ID and attribute certificate can be bundled together, or separated.

- One ID may map to multiple attribute certificates
- Authentication and authorization tradeoffs
  - Efficient and maintenance by multiple CAs
**Binding: Monolithic Signature**

- Single CA for ID and attribute certificate
- Tight binding
  - Every component change needs new issuing
- Simple to implement and be verified, but not flexible
  - Attribute certificate can be changed frequently
  - Not support different lifetime of certificates
  - Not support for multiple attribute CAs

**Binding: Autonomic Signatures**

- Some information in the ID certificate is referred to bind with attribute certificates

---

- Signed by ID CA
Binding: chained Signatures

ID Certificate
- Identity Info.
  - Subject’s Name
  - Subject’s Email
  - Subject Number
- Authentication Info.
  - Subject’s Public-Key Info.
  - Subject’s Password Info.
- Other Info.
  - Serial Number
  - Issue
  - Valid Period
  ETC.
- Signed by ID CA
- ID CA’s Signature

Attribute Certificate
- Binder Info.
  - ID CA’s Signature
- Attributes
  - Role
  - Group
  - ETC.
- Other Info.
  - Serial Number
  - Issuer
  - Valid Period
  ETC.
- Signed by Attribute CA
- Attribute CA’s Signature

Other RBAC Mechanisms
- SAML/XACML (a tentative project topic)
RABC Architecture

Distributed RBAC Architectures
- Server pull architecture
- User pull architecture
- Proxy-based architecture
Server Pull Architecture

Client

Server

User-role Authorization Server

Server-pull Architecture

Role Administrator

User

Role Server

Client (Browser)

Web Server
User Pull Architecture

Client

Server

User-role Authorization Server

User-pull Architecture

Role Assign/Revoke

1: Role Assign/Revoke

1.2: Process Result

Role Server

2.1: Role Request

2.5: Credentials (Roles + Auth-Info*)

2.9: Request Transactions

Web Server

User

Role Administrator

2.3: Role Request

2.7: Validation Request

2.6: Validation Result

User (Browser)

2.8: Request Transactions

2.11: Transaction Results

2.10: Transaction Results
Proxy-based Architecture

Client -> Proxy Server -> Server

User-role Authorization Server

Other RBAC Issues (Topics)

- UA, PA, and RH management
  - By single centralized administrator
  - By general user with administrative permissions
- Attribute based user role assignment
- Delegation
  - User-to-user, user-to-role, role-to-role
- RBAC in workflow control systems
- Inter-organizational RBAC
- And others....