OM-AM Framework

<table>
<thead>
<tr>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectives (policies, requirements, etc)</td>
<td>Assurance</td>
</tr>
<tr>
<td>Models</td>
<td></td>
</tr>
<tr>
<td>Architectures</td>
<td></td>
</tr>
<tr>
<td>Enforcement Mechanisms</td>
<td></td>
</tr>
</tbody>
</table>

Access Control Architecture

- An abstract overview

Access Control Models

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

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>Policy Neutral</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
<td>RBAC96 Model, ARBAC97, etc.</td>
</tr>
<tr>
<td>Architecture</td>
<td>Secure Cookies, Digital Certificates, SAML/XACML, etc.</td>
</tr>
<tr>
<td>How?</td>
<td>RBAC96 System, Assurance</td>
</tr>
<tr>
<td>OM-AM Framework</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
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

RBAC\textsubscript{0} Definition
- \(U, R, P, \text{ and } S\)
  - Users, roles, permissions, sessions
  - \(PA \subseteq P \times R: \text{permission-role assignment}\)
  - \(UA \subseteq U \times R: \text{user-role assignment}\)
  - \(user: S \rightarrow U,\)
  - \(roles: S \rightarrow 2^R,\)
  - \(\text{roles}(s) \subseteq \{r \mid (user(s), r) \in UA\},\)
  - \(Permissions: S \rightarrow 2^P\)
  - \(Permissions(s) \subseteq \bigcup_{r \in \text{roles}(s)} \{p \mid (p, r) \in PA\}\)

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\textsubscript{1}: RBAC\textsubscript{0} + Role Hierarchy

Role Hierarchy: Transitive inheritance of permissions

Primary-Care Physician
  
  Specialist Physician
    
    Physician
      
      Health-Care Provider

Role Hierarchy: Multiple inheritance of permissions

Supervising Engineer
  
  Hardware Engineer
    
    Software Engineer

Role Hierarchy: Private Role

Hardware Engineer
  
  Supervising Engineer
    
    Software Engineer
      
      Engineer

Role Hierarchy Example

Director (DIR)
  
  Engineering Department (ED)
    
    Employee (E)

  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)

    
    Quality Engineer 2 (QE2)
Role Hierarchy Example

Engineer 1 (E1)
- Production Engineer 1 (PE1)
- Quality Engineer 1 (QE1)

Engineer 2 (E2)
- Production Engineer 2 (PE2)
- Quality Engineer 2 (QE2)

Project Lead 1 (PL1)

Project Lead 2 (PL2)

Semantics of RH

- User inheritance
  - r1 ≥ r2 means every user that is a member of r1 is also a member of r2
- Permission inheritance
  - 1 ≥ r2 means every permission that is authorized for r2 is also authorized for r1
- Activation inheritance
  - r1 ≥ r2 means that activating r1 will also activate r2

RBAC1 Definition

- User, Role, Permission, Role-Assignment and User-Role-Assignment (UA) are same as RBAC0.
- RH ⊆ R x R, a partial order with dominance relation ≥.
- roles: S → 2^R,
  - roles(s) ⊆ {r | (∃r')((user(s), r') ∈ UA)}.
- Permissions: S → 2^P
  - Permissions(s) ⊆ U_{r ∈ roles(s)}{p | (∃r'' ≤ r)[(p, r'') ∈ PA]}

RBAC2: RBAC0 + Constraints

RBAC2 Definition

- RBAC2 is unchanged from RBAC0 except for requiring that there be a collection of constraints that determine whether or not values of various components of RBAC0 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.

- **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.

- Conflictions between exclusive roles/permissions
  - When may conflictions happen?
  - How to find conflictions 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 \( A \succeq B \).
  - 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$_2$: RBAC$_0$ + RH + Constraints

OM-AM Framework

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
- Joon S. Park and Ravi Sandhu, Binding Identities and Attributes Using Digitally Signed Certificates, ACSAC 2000

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**

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.

**Attribute Certificate**

- 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**

- Single CA for ID or a multiple CAs for ID
- Simple to implement and be verified, but may not be flexible
- Attribute certificate can be changed frequently
- Not support different lifetimes of certificates
- Not support for multiple attributes

**Binding: Monolithic Signatures**

- Tight binding, every component changes need new issuing
- Simple to implement and be verified, but not flexible
- Attribute certificate can be changed frequently
- Not support different lifetimes of certificates
- Not support for multiple attributes

**Binding: Autonomic Signatures**

- Single signature, change without new issuing
- Simple to implement and be verified, but not flexible
- Not support different lifetimes of certificates
- Not support for multiple attributes

**Bind ID and Attributes**

Attribute Certificate

- Attribute Certificate contains attributes information and is digitally signed.
- Attribute certificate bind subject name to the certificate.
- Attribute authority issues attribute certificate.
- Attribute certificate does not contain public key.
- Attribute certificate needs to be used in conjunction with authentication services, such as another certificate.

**Bind ID and Attributes**

- Attribute certificate binds attribute information to the subject.
- The certificate is digitally signed and issued by an attribute authority.
- Therefore, an attribute certificate needs to be used in conjunction with authentication services, such as another certificate (X.509) and SSL.
Binding: chained Signatures

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

Server-pull Architecture
User Pull Architecture

Client → Server

User-role Authorization Server

User-pull Architecture

Client → Proxy Server → Server

User-role Authorization Server

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....