CS 4910: Intro to Computer Security

Access Control II

Instructor: Xi Tan

Updates

- Lab 1 is due on 2/24
- Assignment 2 is due on 3/05

Review

We already know:

- Access control can be implemented in different ways
  - three important concepts: subject, object, access right
  - access control principles: least privilege, separate of privileges
  - access control matrix, access control lists, capability lists, access control triples
- Discretionary access control (DAC)
  - lets subjects to grant privileges to other subjects at their discretion
- Mandatory access control (MAC)
  - enforces system-wide policy, fixed privileges

Today

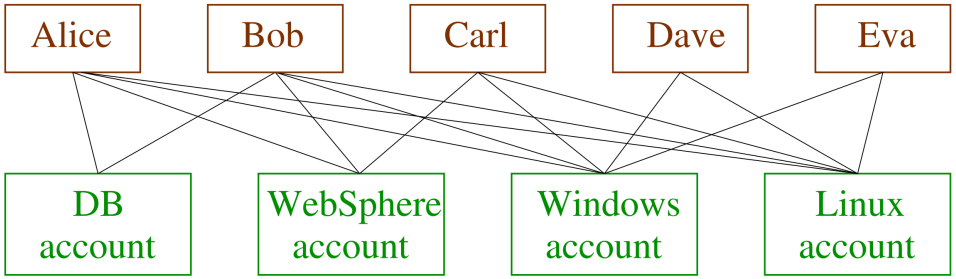
- Role-based access control
- Attribute-based access control

Role-Based Access Control

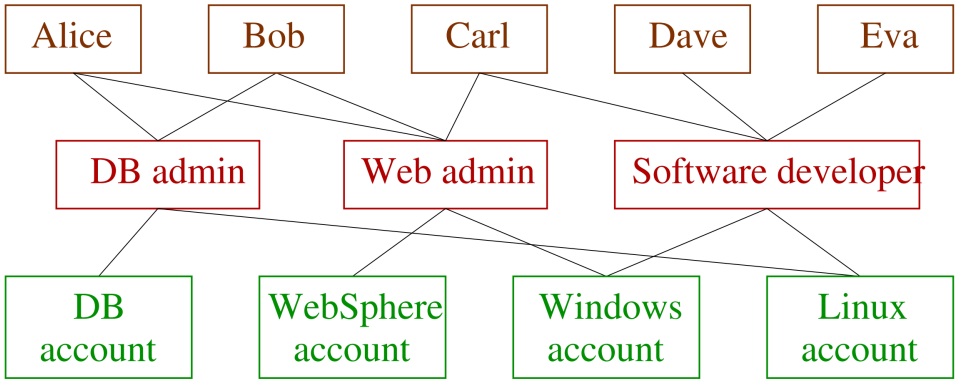
- In Role-Based Access Control (RBAC) models, subjects are combined into "**roles**" according to their privileges in the organization
  - often based on job function
- Permissions are assigned to roles rather than users
- A user can assume **one or more** roles within the organization according to their responsibilities
- RBAC fits operational model of an **organization** and is widely used

Role-Based Access Control

- Non-role-based AC

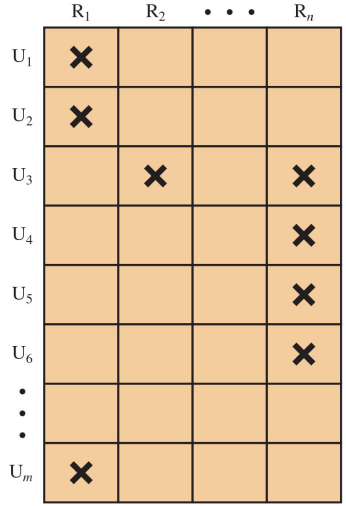


- Role-based AC



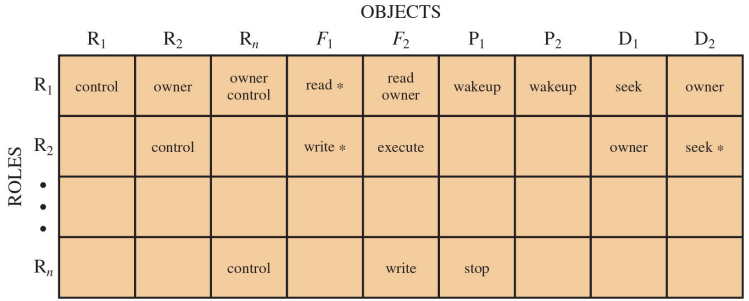
Role-Based Access Control

- Access Control Matrix Representation of RBAC



relates individual users to roles: each matrix entry is either blank or marked, , the latter indicating that this user is assigned to this role

roles as subjects



a role can be treated as an object, allowing the definition of role hierarchies

Role-Based Access Control

- Motivation for RBAC
  - problem: it is difficult to administer user–permission relation
  - roles are a level of indirection
    - “All problems in Computer Science can be solved by another level of indirection” B. Lampson
- RBAC is
  - multi-faceted
  - multi-dimensional
  - open ended
  - ranging from simple to sophisticated

Role-Based Access Control

- Why use roles?
  - fewer relationships to manage
    - potential decrease from O(mn) to O(m + n), where m is the number of users and n is the number of permissions
    - there are often more users than roles and more objects than roles
  - roles are a useful level of abstraction
  - organizations operate based on roles
  - roles are likely to be more stable than the set of users and the set of resources
  - roles can effectively implement the principle of least privilege
    - finding the minimum set of necessary access rights is performed per role rather than per subject

Groups vs. Roles

- How are roles different from groups?
  - A group is a collection of **users**, rather than a collection of **permissions**.
  - Another aspect of RBAC that distinguishes it from traditional group mechanisms is the concept of a session, which is a period of time that allows activation of a subset of roles assigned to a user.

- Paper published in 1996

- By Professor Ravi Sandhu

Feature

Role-Based Access Control Models

Ravi S. Sandhu
George Mason University and SETA Corporation

Edward J. Coyne
Hal L. Feinstein
Charles E. Youman
SETA Corporation

Starting in the 1970s, computer systems featured multiple applications and served multiple users, leading to heightened awareness of data security issues. System administrators and software developers alike focused on different kinds of access control to ensure that only authorized users were given access to certain data or resources. One kind of access control that emerged is role-based access control (RBAC).

A role is chiefly a semantic construct forming the basis of access control policy. With RBAC, system administrators create roles according to the job functions performed in a company or organization, grant permissions (access authorization) to those roles, and then assign users to the roles on the basis of their specific job responsibilities and qualifications (see sidebar "Role-based access control terms and concepts").

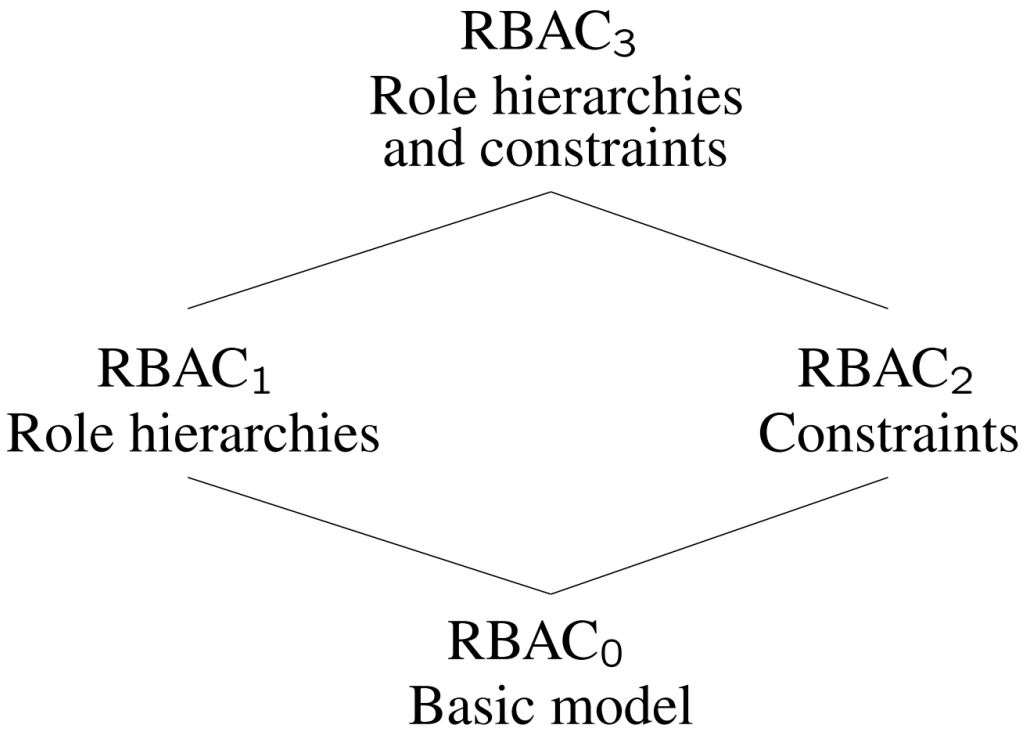
A role can represent specific task competency, such as that of a physician or a pharmacist. A role can embody the authority and responsibility of, say, a project supervisor. Authority and responsibility are distinct from competency. A person may be competent to manage several departments but have the responsibility for only the department actually managed. Roles can also reflect specific duty assignments rotated through multiple users—for example, a duty physician or a shift manager. RBAC models and implementations should conveniently accommodate all these manifestations of the role concept.

Roles define both the specific individuals allowed to access resources and the extent to which resources are accessed. For example, an operator role might access all computer resources but not change access permissions; a security-officer role might change permissions but have no access to resources; and an auditor role might access only audit trails. Roles are used for system administration in such network operating systems as Novell's NetWare and Microsoft's Windows NT.

The particular combination of users and permissions brought together by a role tend to change over time. The permissions associated with a role, on the other hand, are more stable; they tend to change less often than the people who fill the job function that role represents. Therefore, basing security administration on roles rather than on permissions is simpler. Users can be easily reassigned to different roles as needs change. Similarly, as a company acquires new applications and systems, roles can have new permissions granted and existing permissions revoked.

RBAC Models

- The family of RBAC models proposed by Sandhu et al. (1996)

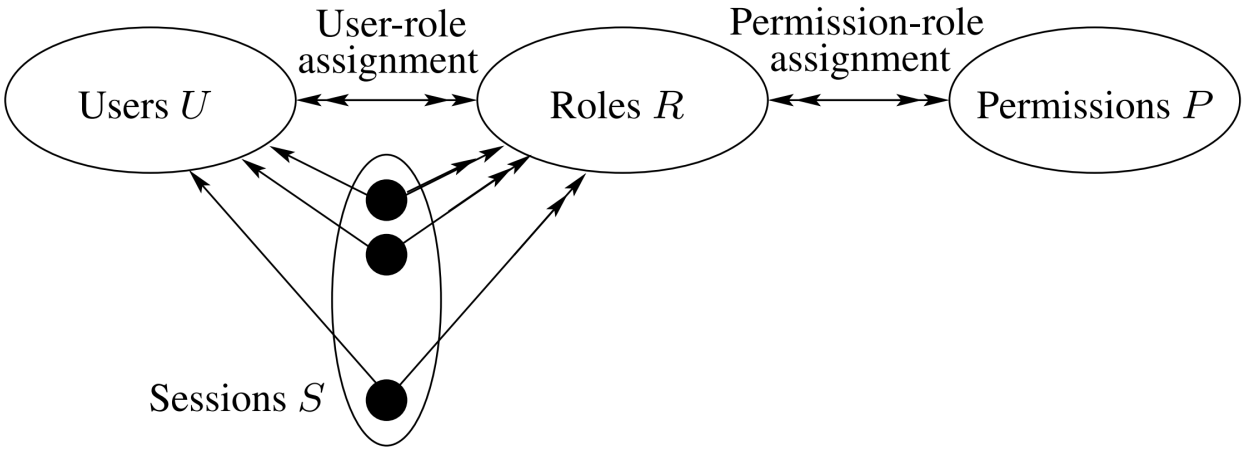


$RBAC_0$

- $RBAC_0$ contains four types of entities
  - users U
  - roles R
  - permissions P
  - sessions S
- User assignment (UA) is many-to-many $UA \subseteq U \times R$
- Permission assignment is many-to-many $PA \subseteq P \times R$
- Session activation
  - one-to-one for user: $S \rightarrow U$
  - one-to-many for roles: $S \rightarrow 2^R$

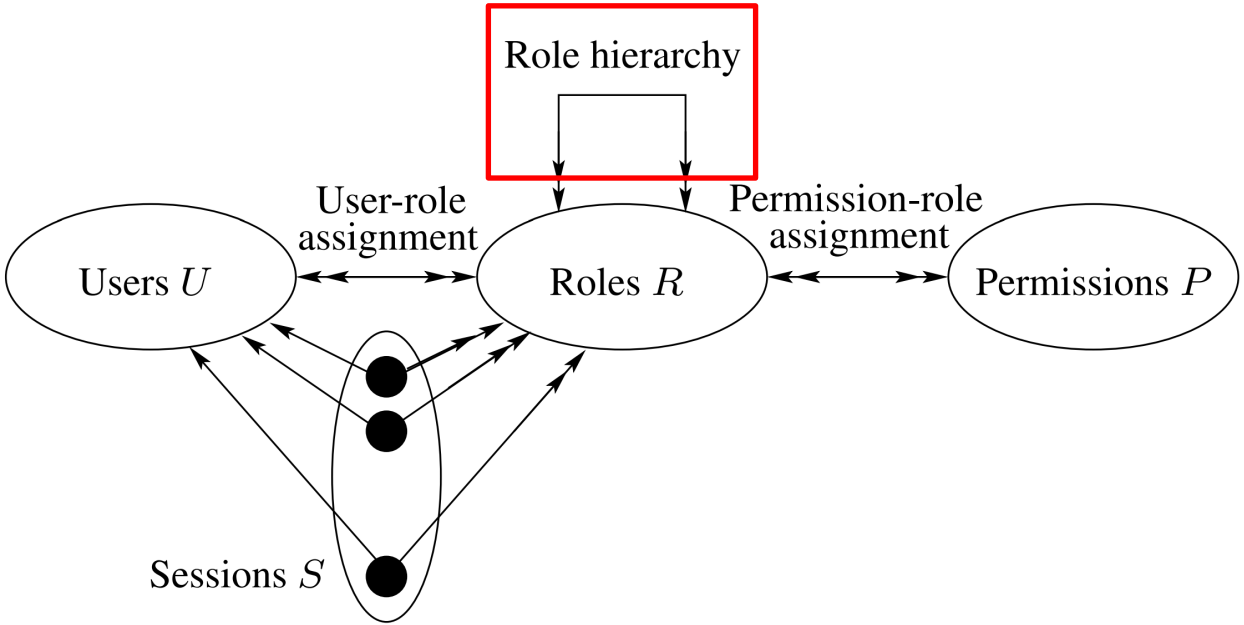
$RBAC_0$

- A session s must comply with UA and PA assignments
  - $roles(s) \subseteq \{r \mid (user(s), r) \in UA\}$
  - permissions of session s are $\cup_{r \in roles(s)} \{p \mid (p, r) \in PA\}$



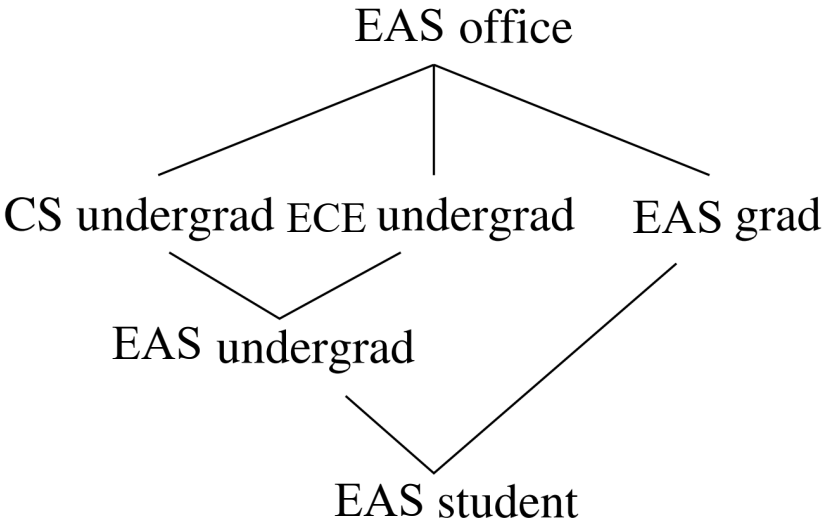
$RBAC_1$

- $RBAC_1$ enhances $RBAC_0$ with role hierarchies



$RBAC_1$

- Role hierarchies are based on the idea that subordinate job functions may have a subset of access rights of a superior job function
  - a role inherits access rights of its descendant roles
- Example of a role hierarchy



$RBAC_1$

- Formal model:
  - U, P, R, S, PA, UA are unchanged from $RBAC_0$
  - role hierarchy $RH \subseteq R \times R$ is a partial order on R whiten as ≥
    - $r_1 \geq r_2$ means that r_1 is an ancestor of r_2
    - partial order means that relationship between any two roles can be undefined
  - requirements on session activation change
    - $roles(s) \subseteq \{r \mid \exists r' \text{ s.t. } [(r' \geq r) \ \& \ (user(s), r') \in UA]\}$
    - session s has permissions

      $\cup_{r \in roles(s)} \{p \mid \exists r' \text{ s.t. } [(r \geq r') \ \& \ (p, r') \in PA\ ]\}$

$RBAC_2$

- No formal model is specified for $RBAC_2$ that adds constraints to $RBAC_0$
- A constraint is a **condition** related to roles or a relationship defined on roles
- Types of constraints (Sandhu et al. 96)
  - mutually exclusive roles
  - cardinality constraints
  - prerequisite constraints

Constraints in RBAC

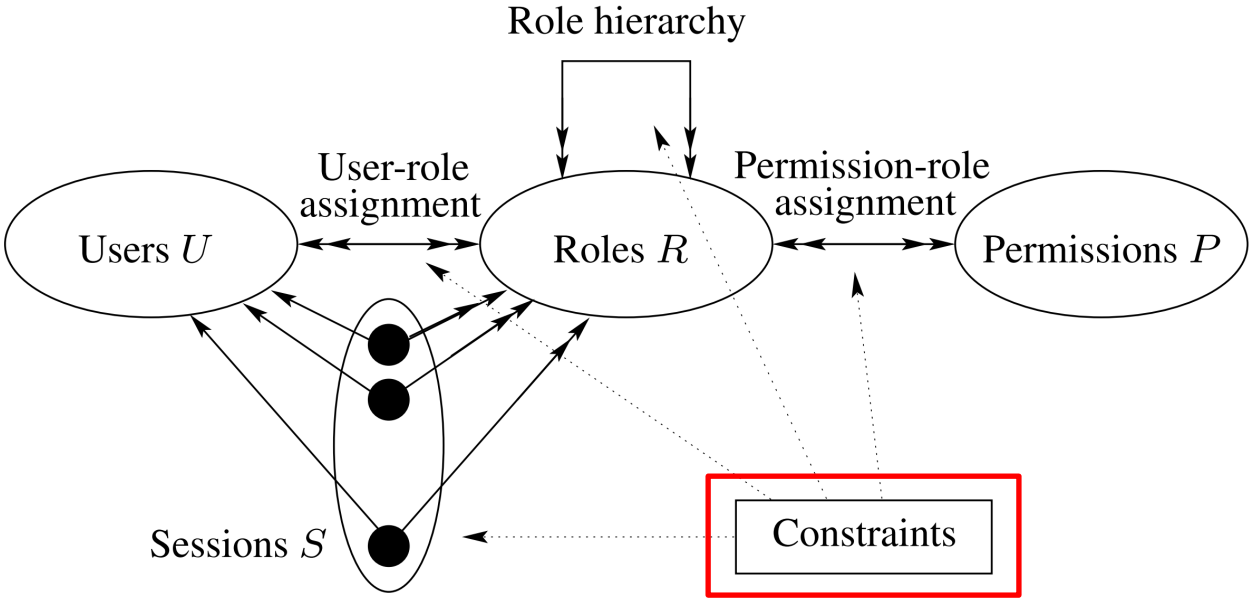
- Mutually exclusive roles: a user can be assigned to only one role from a particular set of roles
  - static exclusion
  - dynamic exclusion
  - such constraints support the separation of duties principle

- Cardinality constraints: setting restrictions on the number of roles

- Prerequisite (or precondition) constraints: the prerequisite must be true before a user can be assigned to a particular role
  - a user can be assigned to role r_1 only if it is already assigned to another role r_2

$RBAC_3$

- $RBAC_3$: features of $RBAC_0$, $RBAC_1$, and $RBAC_2$



- Now role constraints can be based on the role hierarchy

Scope RBAC Models

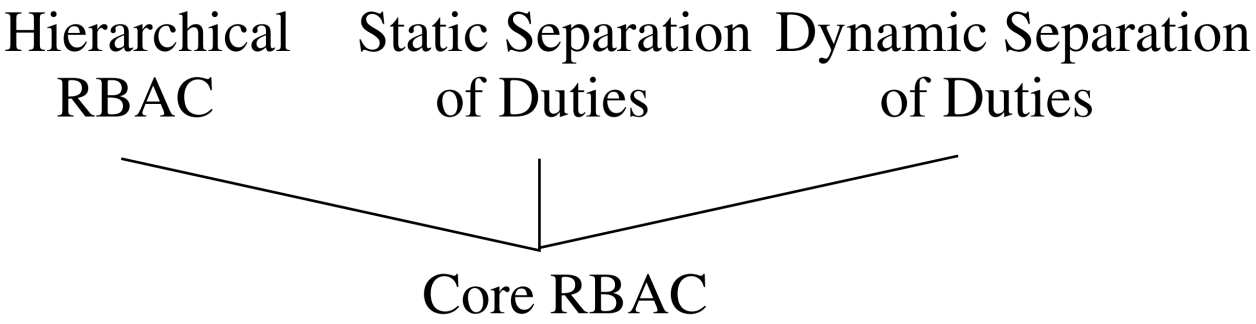
Models	Role Hierarchies	Constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes

RBAC in Use

- Products that use RBAC
  - database management systems (e.g., Oracle)
  - enterprise security management (e.g., IBM Tivoli Identity Manager)
  - operating systems (e.g., Solaris OS, AIX)
- RBAC economic impact study (2002)
  - was conducted by the Research Triangle Institute (RTI) based on interviews with software developers and companies that use RBAC
  - it estimated by 2006 30–50% of employees in service sector would be managed by RBAC systems (10–25% for non-service sectors)
  - it conservatively estimated the economic benefits of this degree of penetration through 2006 to be $671 million

The RBAC Standard

- In 2001 RBAC was proposed to become a NIST standard
- It was adopted as ANSI (American National Standards Institute) standard 359 in 2004
- The standard has the following structure



The RBAC Standard

- The ANSI standard has been criticized by Li et al. (2007)
  - there are many errors
  - there are other limitations and design flaws
  - the publication proposes several changes to the standard
- It was republished as 359-2012 and since reaffirmed as 359-2017 (R2017)
  - the current version consists of two parts: the RBAC reference model and the RBAC system and administrative functional specification

RBAC Extensions

- RBAC has been extensively studied
  - many extensions exist (temporal, geo-spatial, privacy-aware)
  - administration of RBAC
  - constraints, workflow, role engineering, . . .

Next

- Access control principles
  - access control matrices
  - access control lists
  - capability tickets
- Types of access control
  - discretionary access control (DAC)
  - mandatory access control (MAC)
  - role-based access control (RBAC)
  - attribute-based access control (ABAC)

Attribute-Based Access Control

- Attribute-based access control (ABAC) is a rather recent mechanism for specifying and enforcing access control
  - properties are specified in the form of attributes
  - authorizations involve evaluating predicates on attributes
  - conditions on properties of both the subject and resource can be enforced

Attribute-Based Access Control

- ABAC provides a lot of flexibility in specifying rules and supports fine-grained access control
  - it is capable of enforcing DAC, MAC, and RBAC concepts
- This comes at a performance cost
  - it has seen the most success for web services and cloud computing where there is already a response delay
- There are three key elements in an ABAC model
  - attributes
  - policies
  - architecture

Attribute-Based Access Control

- ABAC attributes are characteristics of subjects, objects, environment, and operations preassigned by an authority
- An ABAC model can have three types of attributes
  - subject attributes
    - e.g., name, ID, job function, etc.
  - object attributes
    - e.g., name/title, creation time, ownership information, etc.
  - environment attributes
    - e.g., current date and time, network's security level, etc.

Attribute-Based Access Control

- ABAC architecture specifies how access control is enforced
- When a user submits an access request, the authorization decision is governed by
  - access control policies
  - subject attributes
  - object attributes
  - environmental attributes
- Contrast the above with ACLs in DAC
- Allows an unlimited number of attributes to be combined to satisfy any access control rule

Attribute-Based Access Control

- An ABAC policy is a set of rules and relationships that govern allowable behavior within an organization, based on the **privileges** of subjects and how resources or objects are to be **protected** under which environment conditions
  - Typically written from the perspective of the object that needs protecting and the privileges available to subjects
- Privileges represent the authorized behavior of a subject and are defined by an authority and embodied in a policy
  - Other terms commonly used instead of privileges are: rights, authorizations, and entitlements

Attribute-Based Access Control

- ABAC policies rules implement authorizations using subject-object-environment information (s, o, e)
  - there may not be explicit roles or groups and authorization decisions are instead made based on attributes
  - e.g., consider access to a database of movies
    - everyone can access movies rated as G
    - users of age ≥ 13 can access moved rated as PG-13
    - users of age ≥ 17 can access movies rated as R
    - a policy might be written as P1(s, o, e):

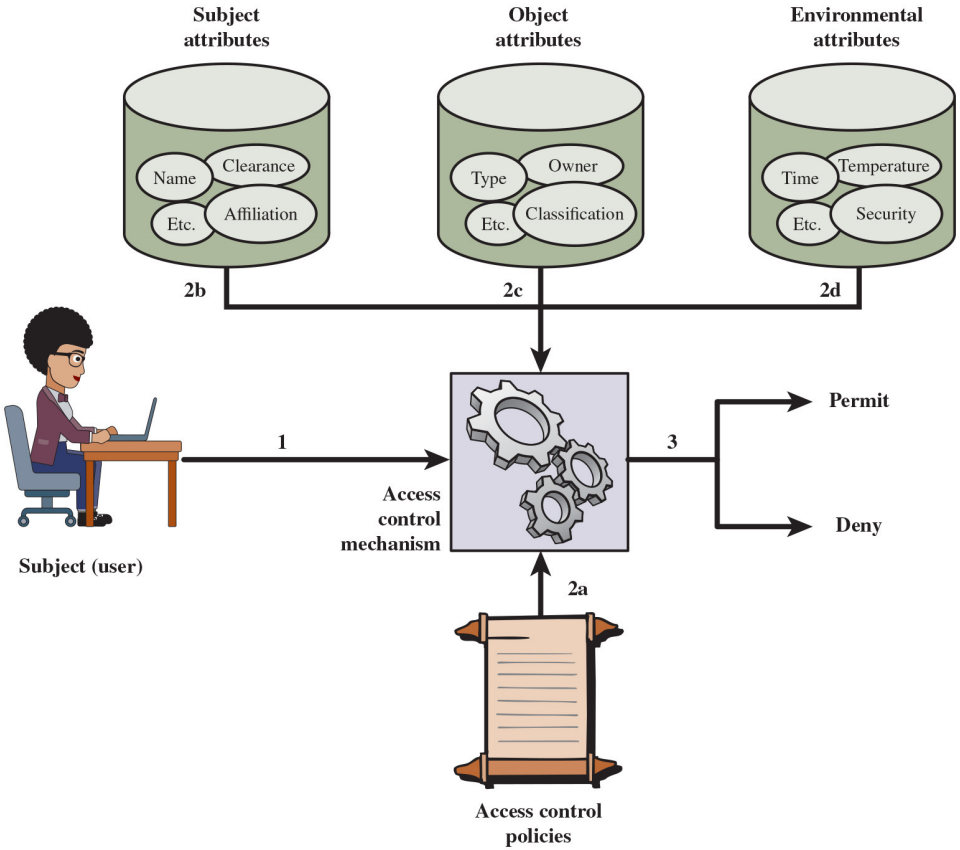
      return (Age(s) ≥ 17 ∧ Rating(o) ∈ {R, PG-13, G}) ∨ (13 ≤ Age(s) < 17 ∧ Rating(o) ∈ {PG-13, G}) ∨ (Age(s) < 13 ∧ Rating(o) ∈ {G})

Attribute-Based Access Control

- ABAC policies can be combined into more complex rules
  - e.g., limit access to new releases to premium membership
    - $P_2(s, o, e)$: return (MemberType(s) = Premium) ∨ (MemberType(s) = Regular ∧ MovieType(o) = OldRelease)
  - grant access if both rules are met
    - $P_3(s, o, e)$: return $P_1(s, o, e) \wedge P_2(s, o, e)$
  - the environment (e.g., the date) can be used for policies such as promotions

Attribute-Based Access Control

- ABAC scenario



Identity Management

- Identity management is related, but not identical to access control
  - it refers to maintaining identity independent of one's job title, job duties, access privileges, location, etc.
  - contrast this with accounts to login into applications, networks, etc.
- A digital identity is typically established based on a set of attributes
  - the attributes together comprise a unique user within a system or enterprise
  - credentials get associated with an identity
  - access is based on credentials that an identity possesses

Identity Management

- Can you use identities maintained by one organization to access systems maintained by other organizations?
  - identity federation refers to the technology, policies and processes to enable this functionality
  - it answers this question via trust
- When disclosing an identity's attributes and credentials to external parties, we generally want to follow the need-to-know principle
- Traditionally identities were maintained by identity service providers which relying parties can use
- More recently, trust network providers regulate interactions between identity service providers and relying parties

Identity Management

Identity Management Applications:

- OpenID is an open standard that allows users to be authenticated by relying parties using third party OpenID identity providers
- Open Identity Trust Framework (OITF) is a standardized specification of a trust framework for identity and attribute exchange
  - it was developed by the community and nonprofit organizations
- Attribute Exchange Network (AXN) is an online gateway for identity service providers and relying parties to access verified identity attributes

Summary

- The choice of an access control model depends on the context
  - system requirements, security policies, etc.
  - can use DAC, MAC, RBAC, ABAC, or other solutions
  - have to consider costs of implementation, maintenance, and rule enforcement

- Federated identity allows for identity credentials to be used across different organizations