Access control can be implemented in different ways:

- **Discretionary access control**
  - lets subjects grant privileges to other subjects at their discretion

- **Mandatory access control**
  - enforces system-wide policy

- **Role-based access control**

- **Attribute-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

- 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
• How are roles different from groups?
  – Answer 1:
  – Answer 2:
  – Answer 3:
• The family of RBAC models proposed by Sandhu et al. (1996)

\[
\begin{align*}
\text{RBAC}_3 & \quad \text{Role hierarchies and constraints} \\
\text{RBAC}_1 & \quad \text{Role hierarchies} \\
\text{RBAC}_0 & \quad \text{Basic model} \\
\text{RBAC}_2 & \quad \text{Constraints}
\end{align*}
\]
**RBAC\(_0\)**

- **RBAC\(_0\)** contains four types of entities
  - users \(U\)
  - roles \(R\)
  - permissions \(P\)
  - sessions \(S\)

- User assignment 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: \(S \rightarrow 2^R\)
• A session $s$ must comply with $UA$ and $PA$ assignments
  
  - $\text{roles}(s) \subseteq \{r \mid (\text{user}(s), r) \in UA\}$
  
  - permissions of session $s$ are $\bigcup_{r \in \text{roles}(s)} \{p \mid (p, r) \in PA\}$
• $\text{RBAC}_1$ enhances $\text{RBAC}_0$ with role hierarchies
• 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

```
CSE office
  └── CS undergrad
      └── CSE undergrad
        └── CSE student
  └── CE undergrad
  └── CSE grad
```
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 $\geq$
  - $r_1 \geq r_2$ 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
    $\bigcup_{r \in roles(s)} \{p \mid \exists r' \text{ s.t. } [(r \geq r') \& (p, r') \in PA]\}$
• 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
• **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

• **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 $r_2$
• **Cardinality constraints**: setting restrictions on the number of roles
  – user-role assignment
    • at most $k$ users can be assigned to the role
    • a user can be assigned to at most $m$ roles
  – role-permission assignment
  – role activation

• **Why should we bother to specify constraints?**
• **RBAC\textsubscript{3}:** features of RBAC\textsubscript{0}, RBAC\textsubscript{1}, and RBAC\textsubscript{2}

> Now role constraints can be based on the role hierarchy
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
Another analysis was performed in 2010

- RBAC use rose to 41% in 2009 and was estimated to be just over 50% in 2010
- over 80% of respondents reported that using roles improved efficiency of maintaining their organization’s access control policy
- economic benefits of RBAC adoption between 1994 and 2009 were estimated at $6 billion
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

```
 Hierarchical RBAC  Static Separation of Duties  Dynamic Separation of Duties
                  |                                |
                  |                                | Core RBAC
```

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

• ABAC systems are thus significantly more complex
• 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 \(\geq 13\) can access movies rated as PG-13
    
    • users of age \(\geq 17\) can access movies rated as R
    
    • a policy might be written as \(P_1(s, o, e)\):
      
      \[
      \text{return } (\text{Age}(s) \geq 17 \land \text{Rating}(o) \in \{R, \text{PG-13}, G\}) \lor (13 \leq \text{Age}(s) < 17 \land \text{Rating}(o) \in \{\text{PG-13}, G\}) \lor (\text{Age}(s) < 13 \land \text{Rating}(o) \in \{G\})
      \]
• 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 \((\text{MemberType}(s) = \text{Premium}) \lor \)
    \((\text{MemberType}(s) = \text{Regular} \land \text{MovieType}(o) = \text{OldRelease})\)

  – grant access if both rules are met

  • \( P_3(s, o, e) \): return \( P_1(s, o, e) \land P_2(s, o, e) \)

  – the environment (e.g., the date) can be used for policies such as promotions
• **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
• **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
• The choice of an access control model depends on the context
  – system requirements, security policies, etc.
  – can use DAC, MAC, RBAC, attribute-based AC, 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