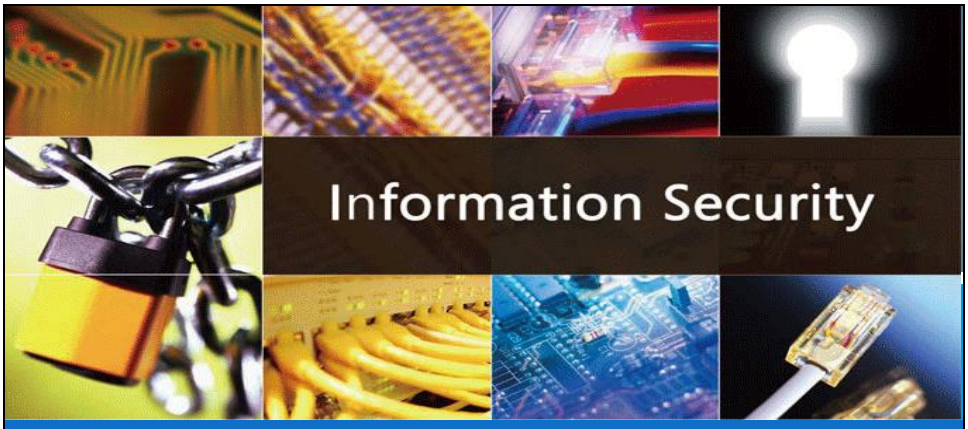


Access control

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Objective

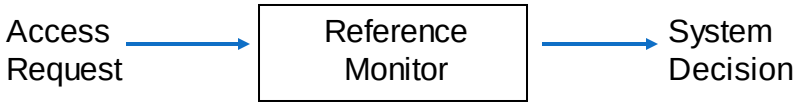
- Introduction to access control and access control structures
- ACL and Capability lists
- Administration and aggregation of access control structures
- BRAC Models
- ACL in Linux
- ACL in Windows

What is access control?

- The process:
  - a computer system controls the interaction between users and system resources
- To implement a security policy, which may be determined by
  - organisational requirements
  - statutory requirements (ex, medical records)
- Policy requirements may include
  - confidentiality (restrictions on read access)
  - integrity (restrictions on write access)
  - availability

A schematic view

- A user requests access (read, write, print, etc.) to a resource in the computer system
- The *reference monitor*
  - establishes the validity of the request …
  - … and returns a decision either granting or denying access to the user



- Ex: RM
  - a paper-based office: the set of (locked) filing cabinets
  - a night club: the security guard + the guest list

Subjects, objects, principal

- U- Subject (user): Active entity in a computer system
  - User, process, thread
- O- Object: Passive entity or resource in a computer system
  - Files, directories, printers
- A *principal*: an attribute or property associated with a subject
  - User ID, Public key, Process, Thread
- Principal and subject: used to refer to the active entity in an access operation
- A subject may be represented by more than one principal

Controlling Accesses to Resources

- **Access Control:** who is allowed to access what.
- Two parts
  - **Part I:** Decide who should have access to certain resources (access control policy)
  - **Part II:** Enforcement – only accesses defined by the access control policy are granted.
- **Complete mediation** is essential for successful enforcement

Access Control Matrix (ACM)

- Introduced by Lampson (1972) and extended by Harrison, Ruzzo and Ullman (1976-8)
- An access control matrix (ACM) **abstracts the state relevant to access control.**
- Rows of ACM correspond to users/subjects/groups
- Columns correspond to resources that need to be protected.
- ACM defines who can access what
  - ACM [U,O] define what access rights user U has for object O.

The access control matrix

Objects / Subjects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}

- The request (jason, allfiles.txt, w) is granted
- The request (mick, allfiles.txt, w) is denied

Disadvantages

- Abstract formulation of access control
- Not suitable for direct implementation
  - The matrix is likely to be extremely sparse and therefore implementation is inefficient
  - Management of the matrix is likely to be extremely difficult if there are 0000s of files and 00s of users (resulting in 000000s of matrix entries)

Access control lists

- Access control lists focus on the objects
  - Typically implemented at operating system level
  - Windows NT uses ACLs
  - an ACL be stored In trusted part of the system
- An ACL corresponds to a **column** in the access control matrix

Ex: [a.out: (jason, {r,w,x}), (mick, {r,x})]

- How would a reference monitor that uses ACLs check the validity of the request (jason, a.out, r)?

Objects / Subjects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}

Capability lists

- A capability list corresponds to a **row** in the access control matrix

Ex [jason: (trash, {r,w}), (a.out, {r,w,x}), (allfiles.txt, {r,w})]

- How would such a reference monitor check the validity of the request (jason, a.out, r)?

Subjects \ Objects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}

Capability lists

- Where do C-lists go?
  - User catalogue of capabilities defines what a certain user can access
  - Can be stored in objects/resources themselves (Hydra)
  - Sharing requires propagation of capabilities
- Capability lists focus on the subjects
  - in services and application software
  - Database applications: use capability lists to implement fine-grained access to tables and queries
  - Renewed interest in capability-based access control for distributed systems
- Disdavantage
  - How can we check which subjects can access a given object ("before-the-act per-object review")?

Administration

- Tasks include
  - Creation of new objects and subjects
  - Deletion of objects and subjects
  - Changing entries in access control matrix (changing entries in ACLs and capability lists)
- The administration of access control structures is extremely time-consuming, complicated and error-prone
- To simplify the administrative burden: AC structures that aggregate subjects and objects are used
- Aggregation techniques
  - User groups
  - Roles
  - Procedures
  - Data types

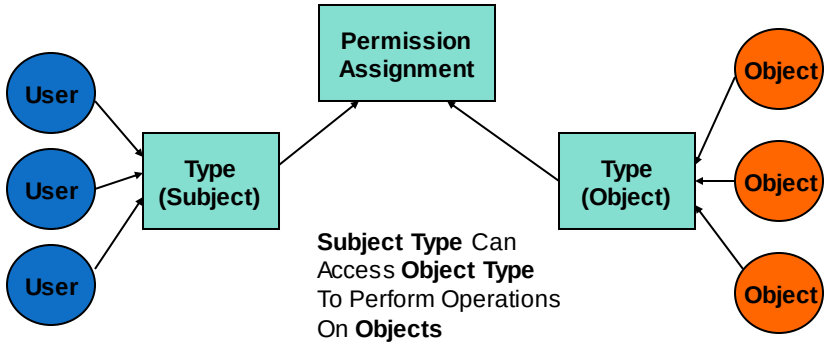
Groups

- Access rights are often defined for groups of users
  - In UNIX three groups are associated with each object
    - Owner
    - Group (owner)
    - Others
  - In VMS there are four groups
    - Owner
    - Group
    - World
    - System

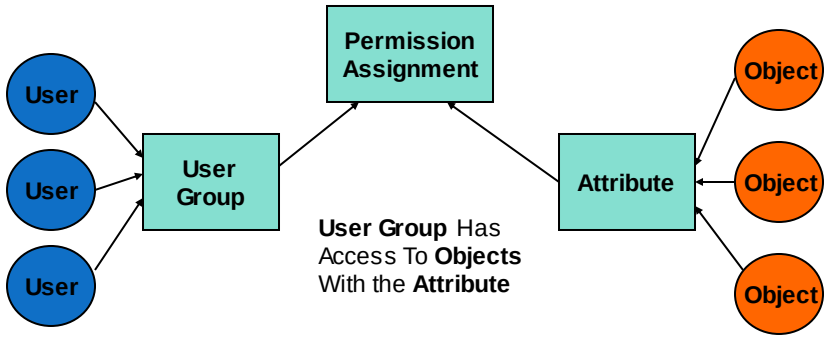
Roles

- A *data type* is a set of objects with the same structure (bank accounts, for example)
- We define access operations (*procedures* or *permissions*) on a data type
- Permissions are assigned to roles
- Users are assigned to roles
- Roles are (usually) arranged in a hierarchy

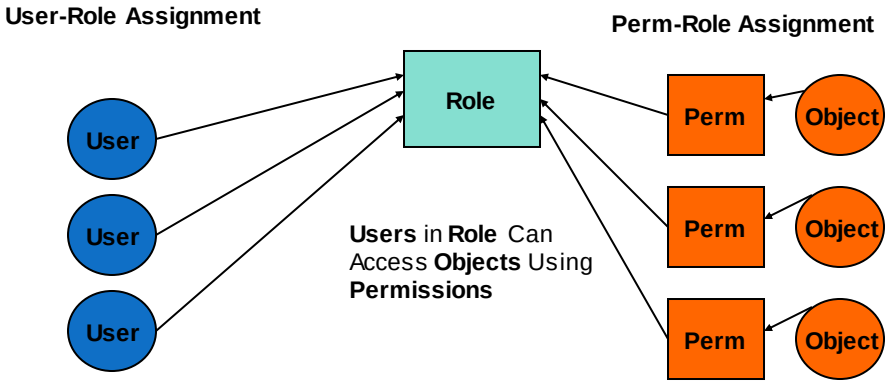
Type Enforcement [BoebertKain84]



Group and Attributes



Role-based Access Control



Role-based Access Control Model

- Users: U
- Permissions: P
- Roles: R
- Assignments: User-role, perm-role, **role-role**
- **Sessions: S**
- Function: user(S), roles(S)
- **Constraints: C**
- Type Enforcement
  - E: set of subjects or objects
  - Permission Assignment
    - ST: set of subject types
    - OT: set of object types
    - O: set of operations

Access Control

- ACL is used by many OS to determine whether users are authorized to conduct different actions
  - the mandatory access control (MAC): computer system

    the computer system decides exactly who has access to which resource in the system
  - the discretionary access control (DAC): users

    users are authorized to determine which other users can access files or other resources that they create
  - the role-based access control (RBAC): MAC in special

    the system decides exactly which users are allowed to access which resources—but the system does this in a special way
- The Bell-LaPadula Model: certain level of access.

Access Control

ACL

MAC

Role	Resource	Privilege
Backup Operator	/home/*	Read
Administrator	/*	Read, write, execute

DAC

User	Resource	Privilege
Alice	/home/Alice/*	Read, write, execute
Bob	/home/Bob/*	Read, write, execute
*	/home/Alice/product_specs.txt	Read

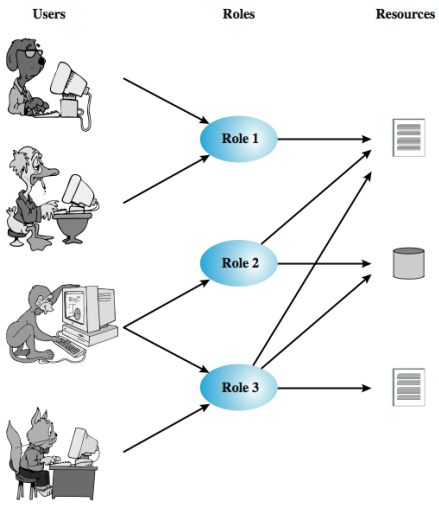
BRAC

Role	Resource	Privilege
Backup Operator	/home/*	Read
Administrator	/*	Read, write, execute
Programmer	/home/Alice/product_specs.txt	Read

BRAC Model

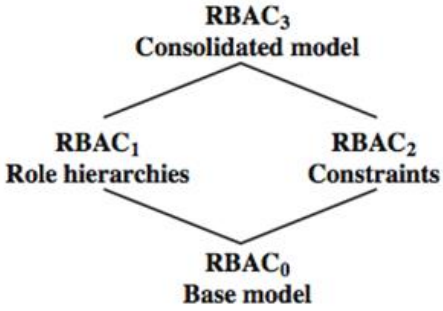
- BRAC Model:

the system decides exactly which users are allowed to access which resources—but the system does this in a special way



BRAC Model

- $RBAC_0$: the minimum functionality
- $RBAC_1$: the $RBAC_0$ functionality + role hierarchies, which enable one role to inherit permissions from another role.
- $RBAC_2$: $RBAC_0$ + constraints, which restricts the ways in which the components of a RBAC systemmay be configured.
- $RBAC_3$: $RBAC_0$ + $RBAC_1$ + $RBAC_2$

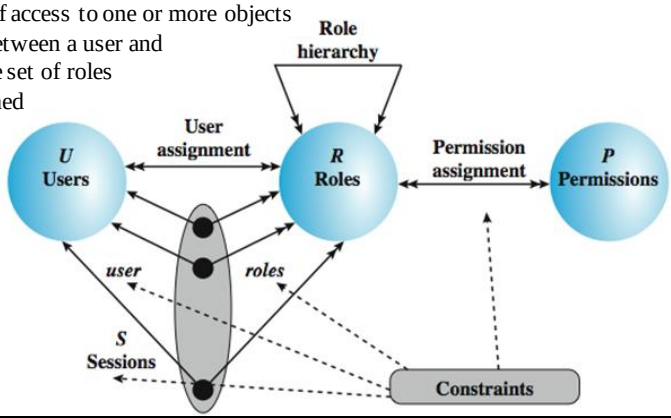


(a) Relationship among RBAC models

BRAC Model

- An $RBAC_0$ systemcontains the four types of entities (the minimum functionality for an RBAC system):

- **User:** An individuals - access to this computer system
- **Role:** job function - controls this computer system
- **Permission:** approval of access to one or more objects
- **Session: :** A mapping between a user and an activated subset of the set of roles to which the user is assigned



RBAC Family of Models

- $RBAC_0$ contains all but hierarchies and constraints
- $RBAC_1$ contains $RBAC_0$ and hierarchies
- $RBAC_2$ contains $RBAC_0$ and constraints
- $RBAC_3$ contains all
- The RBAC family idea has always been more a NIST initiative
- The RBAC families are present in the NIST RBAC standard [NIST2001] with slight modifications:
  - $RBAC_0$, $RBAC_1$ (options), $RBAC_3$ (SSD) , $RBAC_3$ (DSD)

Benefits of RBAC

- We only need to assign users and permissions to roles
- We can use inheritance in the role hierarchy to reduce the number of assignments that are required
- Simplifies administration

RBAC models

- NIST (Ferraiolo et al., 1992-2000)
- RBAC96 (Sandhu et al., 1996)
- ARBAC97 (Sandhu et al., 1997-99)
- OASIS (Hayton et al., 1996-2001)
- Role Graph model (Nyanchama and Osborn, 1995-2001)
- Unified RBAC96 NIST model (Ferraiolo, Sandhu et al., 2001)

RBAC implementations

- Roles implemented in
  - Window NT (as global and local groups)
  - IBM's OS/400
  - Oracle 8 onwards
  - .NET framework
- There is no generally accepted standard for RBAC
  - Role hierarchies
  - Semantics of role hierarchies

Need for Aggregate Models (RBAC)

- Practical ease of specification
  - Abstraction for users, permissions, constraints, administration
- Natural access control aggregations – based on organizational roles
  - As new employees join, their permission assignments are determined by their job
  - Permission assignment is largely static
- Central control and maintenance of access rights
- Flexible enough to enforce
  - least privilege, separation of duties, etc.

Hierarchies and Constraints

- Role hierarchy
  - Problem: does organizational hierarchy correspond to a permission inheritance hierarchy?
  - Problem: do organizational roles make sense for building hierarchies?
- Constraints
  - Problem: constraints apply to all states, so they require a predicate calculus in general
  - Problem: Only certain types of constraints can effectively be administered? Mutual exclusion, separation of duty, cardinality, etc.
- Conflicts
  - May find other concepts useful for resolving conflicts between constraints and hierarchies/assignments

Does RBAC Achieve Its Goals?

- Practical ease of specification
  - Clear base model – need more help for constraints, admin
- Natural access control aggregations – based on organizational roles
  - In some cases, but not clear that organizational roles help with permission assignment – particularly with inheritance
- Central control and maintenance of access rights
  - Central view is a selling feature of products, but a single view of all can be complex (layering?)
- Flexible enough to enforce
  - Flexible access control expression, but difficult to determine if we enforce our security goals (constraints)

Access Control Implementation in Unixlike Systems

- Each file has an owner, who has a unique user ID (UID).
- Access is possible for an owner, group, and world.
- Permissions are read, write, execute.

- The original ACL implementation had a fixed size representation (9 bits).
- Now full ACL support is available for many variants
  - Setuid this is for users to have write access during a specific period of time.

ACL in Linux

- provide a finer-grained control over which users can access specific directories and files.
- Using ACLs, you can specify the ways in which each of several users and groups can access a directory or file.
- Commands:
  - displays the file name, owner, group and the existing ACL for a file: **getfacl**
  - sets ACLs of files and directories: **setfacl -m**

```
setfacl -m ugo:u/g_name:permissions fil/fol_name
```

  - removes rules in a file or folder's: **setfacl -x**

Use numeric or character to set permission

ACL in Windows

- Commands:
  - List: **net user, net localgroup**
  - Change the permisions: **acls**
  - Testing - quickly start a program as another user: **runas**

Ex, `runas /User:jack cmd.exe`

Q & A