



Information Security

Access control

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Objective

- ✧ Introduction to access control
 - access control structures
 - ACL and Capability lists
 - Administration and aggregation of access control structures
- ✧ AC Policies:
 - MAC, DAC, BRAC, ABAC
- ✧ ACL in Linux
- ✧ ACL in Windows

AAA

AAA is an architectural framework for configuring:

A

Authentication - Who is allowed access?

A

Authorization - What are they allowed to do?

A

Accounting - What did they do?



08/10/2024

"Access control" = "authorization"

3

AAA

Authentication

Who are you?



Authorization

How much can you spend?

Reference Number	Date	Posted	Activity Since Last Statement	Amount
4324567	01-03	01-13	Payment, Thank You	-\$14.25
51234567	01-12	01-13	Wings 'N' Things	\$25.25
78901234	01-14	01-17	Record Release	\$40.00
45678901	01-14	01-17	Sports Stadium	\$75.25
3210987	01-22	01-23	Tie Tack	\$20.75
76543210	01-29	01-30	Electronic World	\$89.25
2345678	01-30	01-30	Transaction Fees	\$3.00
34567890	01-01	01-01	Annual Fee	\$25.00

Accounting

What did you spend it on?

08/10/2024

4

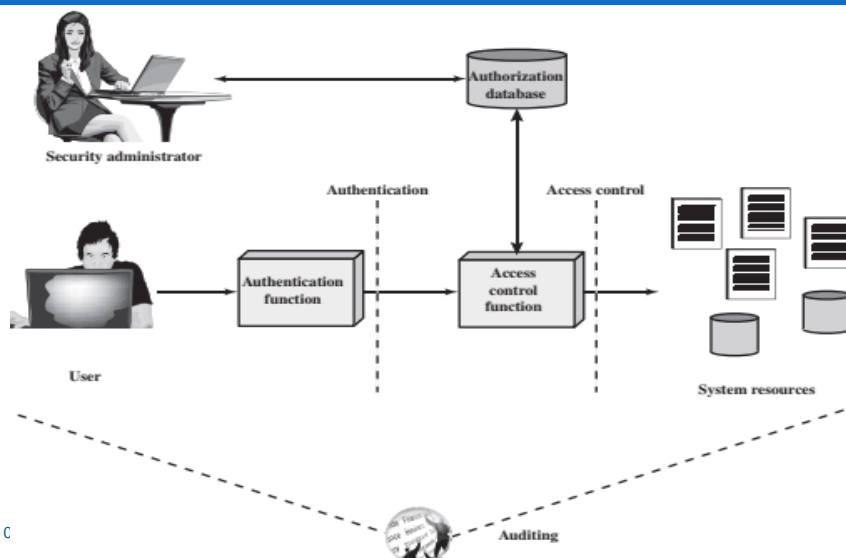
AAA Benefits

- ∞ Increased flexibility and control of access configuration
- ∞ Scalability
 - AAA is typically implemented using a dedicated ACS server to store usernames / passwords in a centralized database.
 - Information is centrally entered / updated unlike a local database which must be configured on every router
- ∞ Multiple backup systems: Fault Tolerance can be configured in a fallback sequence.
 - Consult a security server...
 - If error or none, consult local database,

08/10/2024

5

Relationship Among Access Control and Other Security Functions

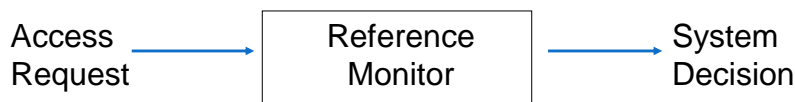


What is access control?

- ∞ 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

Types of access controls

- ✎ **Vertical access controls** are mechanisms that restrict access to sensitive functionality to specific types of users.
 - different types of users have access to different application functions.
 - For example, an administrator might be able to modify or delete any user's account, while an ordinary user has no access to these actions.
- ✎ **Horizontal access controls** are mechanisms that restrict access to resources to specific users.
 - different users have access to a subset of resources of the same type.
 - For example, a banking application will allow a user to view transactions and make payments from their own accounts, but not the accounts of any other user.
- ✎ **Context-dependent access controls** restrict access to functionality and resources based upon the state of the application or the user's interaction with it.
 - prevent a user performing actions in the wrong order.
 - For example, a retail website might prevent users from modifying the contents of their shopping cart after they have made payment.

08/10/2024

11

Broken access control

- ✎ Broken access control vulnerabilities exist when a user can access resources or perform actions that they are not supposed to be able to.
- ✎ **Vertical privilege escalation**
 - For example, if a non-administrative user can gain access to an admin page where they can delete user accounts.
- ✎ **Horizontal privilege escalation**
 - occurs if a user is able to gain access to resources belonging to another user, instead of their own resources of that type.
 - For example, if an employee can access the records of other employees as well as their own

08/10/2024

12

Authorization exists on multiple levels

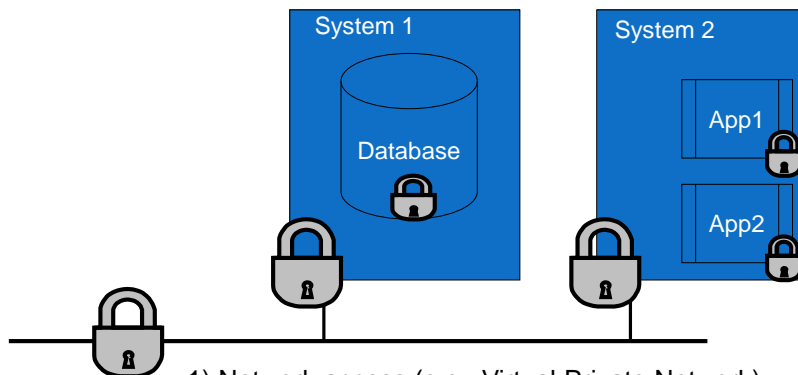
Level	Subject	Action	Guard	Protected System
Hardware	OS Process	Read memory	CPU	CPU and Memory
Network	Host	Send packets	Firewall	Intranet
Database	User	SELECT query	DBMS	User database
OS	User	Open file	OS Kernel	Filesystem
OS	Java Program	Open file	Java Security Manager	Filesystem
Application	User	Read patient file	Application code	Application data

08/10/2024

13

Four Layers of Logical Security

Logical security is security in software, as opposed to physical security



- 1) Network access (e.g., Virtual Private Network)
- 2) Computer access (login/password), MAC
- 3) Database access (permissions), DAC, RBAC
- 4) Application access (permissions), RBAC

System Access Control

- ✎ Establish rules for access to information resources
- ✎ Create/maintain user profiles
- ✎ Allocate user IDs requiring authentication (per person, not group)
- ✎ Notify users of valid use and access before and upon login
- ✎ Ensure accountability and auditability by logging user activities
- ✎ Log events
- ✎ Report access control configuration & logs

Application-Level Access Control

- ✎ Create/change file or database structure
- ✎ Authorize actions at the:
 - Application level
 - File level
 - Transaction level
 - Field level
- ✎ Log network & data access activities to monitor access violations

Elements in Access control

- ✎ **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

Elements in Access control

- ✎ An **access right** describes the way in which a subject may access an object:
 - **Read:** User may view information in a system resource (e.g., a file, selected records in a file, selected fields within a record, or some combination). Read access includes the ability to copy or print.
 - **Write:** User may add, modify, or delete data in system resource (e.g., files, records, programs). Write access includes read access.
 - **Execute:** User may execute specified programs.
 - **Delete:** User may delete certain system resources, such as files or records.
 - **Create:** User may create new files, records, or fields.
 - **Search:** User may list the files in a directory or otherwise search the directory

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



Subjects \ Objects	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)
- ∞ ACM solution:
 - decompose the matrix for ease of storage and retrieval and
 - represent these components using expression concepts
 - ACL
 - Capability list

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)?

Subjects \ Objects	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?

- 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

☞ Disadvantage

- How can we check which subjects can access a given object ("before-the-act per-object review")?

Administration in Access control

☞ 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)

☞ Challenges:

- extremely time-consuming, complicated and error-prone

☞ To simplify the administrative burden: need aggregate subjects and objects are used

☞ Aggregation techniques

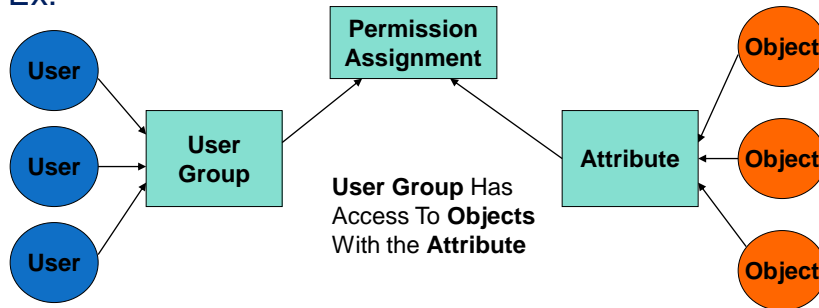
- User groups
- Roles
- Procedures (policies)
- Data types

Aggregation techniques : 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

∞ Ex:

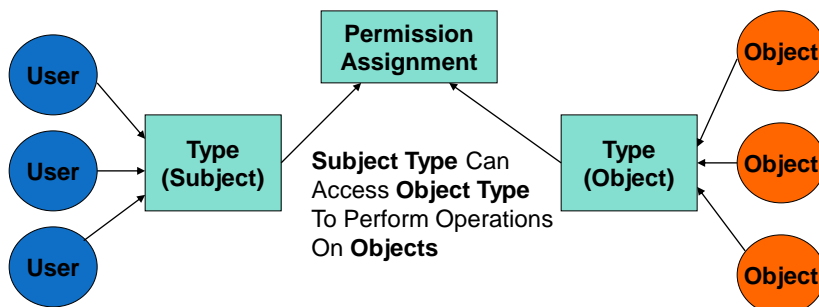


Aggregation techniques Data type, procedures

∞ 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

∞ Ex:



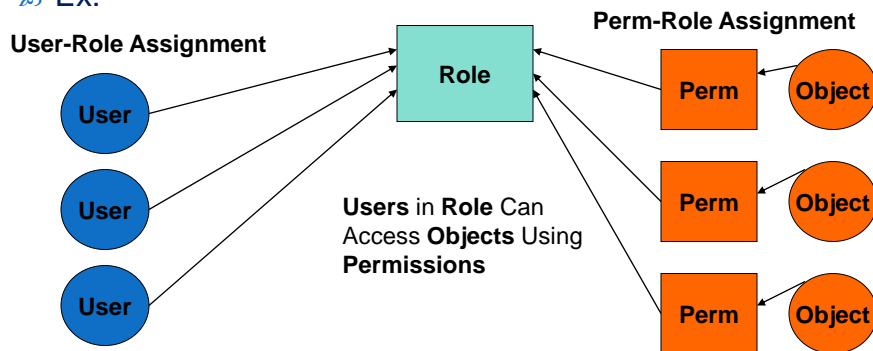
Aggregation techniques

Roles

Role:

- Permissions are assigned to roles
- Users are assigned to roles
- Roles are (usually) arranged in a hierarchy

Ex:



Access Control models

- 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
 - **Attribute-based access control (ABAC):** Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions.
 - **Physical Access Control:** Locks, fences, biometrics, badges, keys
- The Bell-LaPadula Model: certain level of access.



Access Control

ACL, ex

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

RBAC

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

08/10/202

Access Control Techniques

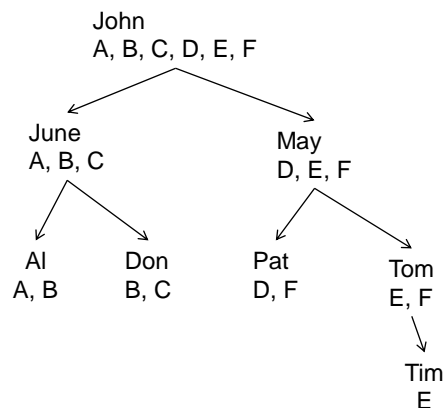
Mandatory Access Control

Login	User	Group	Permi...
John	John	Mgmt	rw x r x
June	June	Billing	r
May	May	Factory	r x r x
Al	Al	Billing	
Don	Don	Billing	

Role-Based Access Control

Login	Role	Permission
John	Mgr	A, B, C, D, E, F
June	Acct.	A, B, C
Al	Acct.	A, B, C
May	Factory	D, E, F
Pat	Factory	D, E, F

Discretionary Access Control



DAC - Discretionary Access Control

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Discretionary Access Control



In discretionary access control (DAC), **owner of a resource decides** how it can be shared

- Owner can choose to give **read or write access to other users**

Discretionary Access Control



Two problems with DAC:

- You cannot control if someone you share a file with will not further share the data contained in it
 - **Cannot control “information flow”**
- In many organizations, **a user does not get to decide how certain type of data can be shared**
 - Typically the employer may mandate how to share various types of sensitive data
 - Mandatory Access Control (MAC) helps address these problems



DAC Quiz

Check the best answer:

In a certain company, payroll data is sensitive. A file that stores payroll data is created by a certain user who is an employee of the company. Access to this file should be controlled with a...

- ☐ DAC policy that allows the user to share it with others carefully
- ☐ It must use a MAC model as the company must decide who can access it

User works in a company and the **company decides how data should be shared**

MAC - Mandatory Access Control

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Mandatory Access Control (MAC) Models

Military and intelligence agencies:



Data has **associated classification level** and users are cleared at various levels

- Top secret, secret, confidential etc.
- Limits on **who can access data at a certain level**
 - User cleared only at secret level should not be able to access top secret data
- Also called **multilevel security (MLS)**

Multi-level Security (MLS)

- ✎ The capability of a computer system to:
 - carry information with different sensitivities (i.e. classified information at different security levels),
 - permit simultaneous access by users with different security clearances and needs-to-know, and
 - prevent users from obtaining access to information for which they lack authorization.
- ✎ Discretionary access control fails to achieve MLS
- ✎ Typically use Mandatory Access Control
 - → Primary Security Goal: Confidentiality

39

Mandatory Access Control

- ✎ Mandatory access controls (MAC) restrict the access of subjects to objects based on a system-wide policy
 - denying users full control over the access to resources that they create.
 - The system security policy (as set by the administrator) entirely determines the access rights granted

Implementing MAC

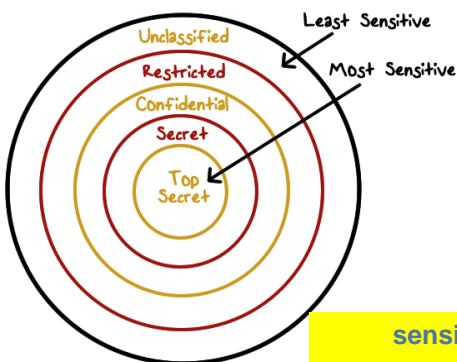
Labels: A Key Requirement for Implementing MAC

- indicate sensitivity/category of data or clearance/need-to-know requirements of users
- TCB associates **labels with each user and object and checks them when access requests are made**
 - Need to relate labels to be able to compare them
- Exact nature of labels **depends on what kind of model/policy is implemented**
 - DoD models include classification/clearance level and a compartment in the label
 - Commercial policies are different but use labels to deal with conflict-of-interest, separation-of-duty etc.



Multilevel in Implementing MAC

Example of Labels/MAC in a DoD Environment:



- **Label** = (sensitivity level, Compartments)
- Let us consider highly sensitive documents that have information about various arms stockpiles.
 - $L1 = (TS, \{\text{nuclear, chemical}\})$
 - $L2 = (S, \{\text{nuclear, conventional}\})$
- Providing confidential access to documents (Bell and La Padula or BLP Model)

sensitivity levels are totally ordered
(TS > S > C > U)

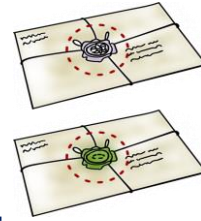
Comparing Labels

- Compartments are sets which can only be partially ordered

- How do we order labels?

$$L_1 = (X_1, \text{Comp}_1); L_2 = (X_2, \text{Comp}_2):$$

- L_1 dominates L_2 or $L_1 > L_2$: $X_1 > X_2$ and $\text{Comp}_1 \geq \text{Comp}_2$
- L_1 is dominated L_2 or $L_1 < L_2$: $X_1 < X_2$ and $\text{Comp}_1 \leq \text{Comp}_2$
- $L_1 = L_2$: $X_1 = X_2$ and $\text{Comp}_1 = \text{Comp}_2$
- L_1 and L_2 are not compatible: $L_1 \neq L_2$



Ordering Among Labels

Ordering among labels defines a structure called a lattice:

Example:

	$L_1 = (TS, \{A,B,C\})$	$L_1 > L_2?$
<u>Partial Order</u>	$L_2 = (S, \{A,B\})$	$L_2 < L_1?$
	$L_3 = (S, \{B,C,D\})$	L_1 and L_3



Label Domination Quiz

Select the best answer:

If $L1 = (\text{secret}, \{\text{Asia}, \text{Europe}\})$ and
 $L2 = \{\text{top-secret}, \{\text{Europe}, \text{South-America}\}\}$,

- ☐ L1 dominates L2
- ☐ L2 dominates L1
- ☐ Neither L1 nor L2 dominates the other one



Sensitive Data Quiz

Select the best answer:

Assume that label L1 or a document D1 dominates label L2 of document D2 when these labels are defined by (sensitivity level, compartment).

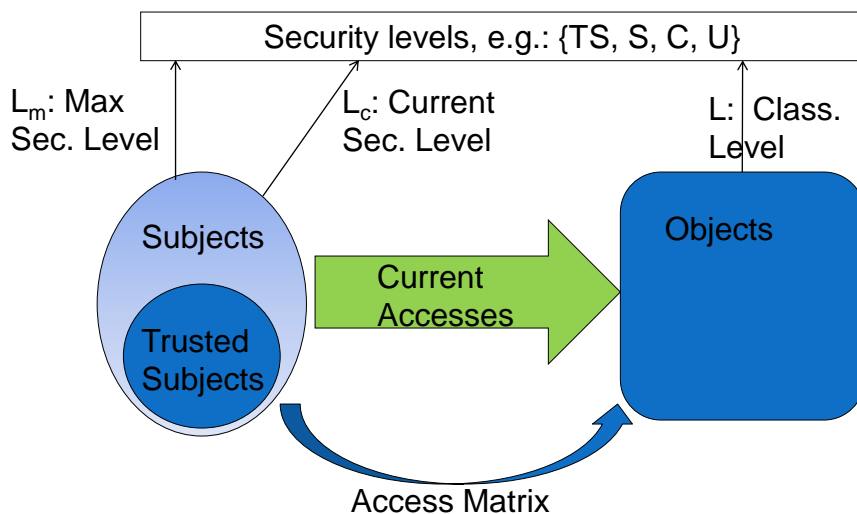
- ☐ D1 contains more sensitive data than D2.
- ☐ D2 is more sensitive than D1.
- ☐ The data contained in D2 has a narrower scope as defined by its compartment

Bell-LaPadula Model: A MAC Model for Achieving Multi-level Security

- ∞ Introduce in 1973
- ∞ Air Force was concerned with security in time-sharing systems
 - Many OS bugs
 - Accidental misuse
- ∞ Main Objective:
 - Enable one to formally show that a computer system can securely process classified information

47

Elements of the BLP Model



48

The BLP Security Model

- ✎ A computer system is modeled as a state-transition system
 - There is a set of subjects; some are designated as **trusted**.
 - Each state has objects, an access matrix, and the current access information.
 - There are state transition rules describing how a system can go from one state to another
 - Each subject s has a maximal sec level $L_m(s)$, and a current sec level $L_c(s)$
 - Each object has a classification level

49

The BLP Security Policy

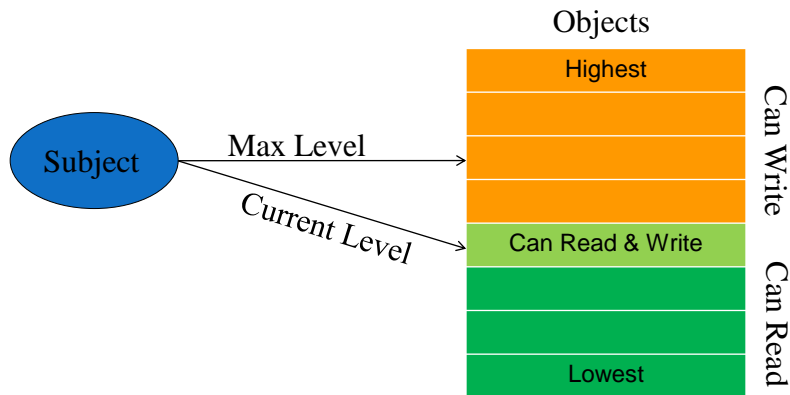
- ✎ A state is secure if it satisfies
 - Simple Security Condition (**no read up**):
 - S can read O iff $L_m(S) \geq L(O)$

User with label L_m can read document with label L only when L_m **dominates** L
 - The Star Property (**no write down**): for any S **that is not trusted**
 - S can read O iff $L_c(S) \geq L(O)$ (no read up)
 - S can write O iff $L_c(S) \leq L(O)$ (no write down)

User with label L_m can write document with label L when L_m **is dominated by** L
 - Strong Star Property
 - S can read and write O iff $L_c(S) = L(O)$
 - Discretionary-security property
 - every access is allowed by the access matrix
- ✎ A system is secure: every reachable state is secure.

50

Implication of the BLP Policy



Read-down rule (simple security property)

User with label L1 can read document with label L2 only when **L1 dominates L2**

Write-up rule (star property)

User with label L1 can write document with label L2 when **L1 is dominated by L2**

Star-property

- ✎ Applies to subjects (principals) not to users
- ✎ Users are trusted (must be trusted) not to disclose secret information outside of the computer system
- ✎ Subjects are not trusted because they may have Trojan Horses embedded in the code they execute
- ✎ Star-property prevents overt leakage of information and does not address the covert channel problem

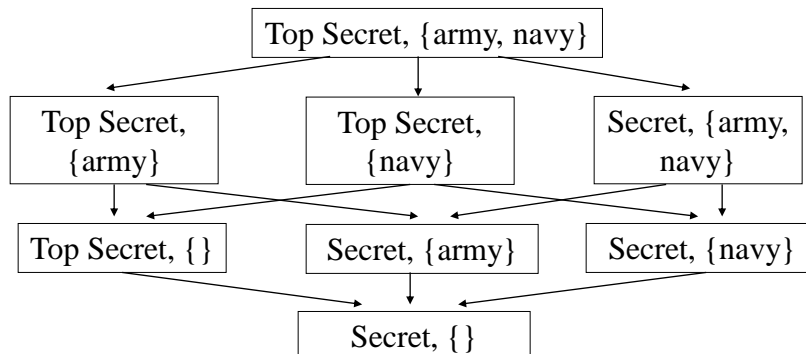
More on MLS: example

- ✎ Used as attributes of both subjects & objects
 - clearance & classification
- ✎ MLS
 - security levels
 - security categories: Also known as compartments
 - security labels = Levels \times P (Categories)
- ✎ Define an ordering relationship among Labels
 - $(e1, C1) \leq (e2, C2)$ iff. $e1 \leq e2$ and $C1 \subseteq C2$

Apply	security levels	security categories
military	top secret \geq secret \geq confidential \geq unclassified	<ul style="list-style-type: none"> • army, navy, air force • nato, nasa, nofor
commercial	restricted \geq proprietary \geq sensitive \geq public	<ul style="list-style-type: none"> • Sales, R&D, HR • Dept A, Dept B, Dept C

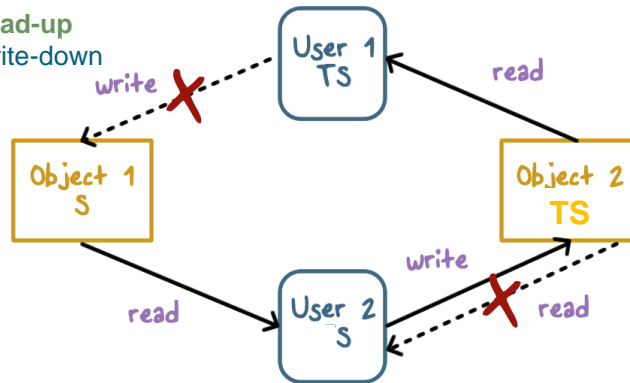
An Example Security Lattice

- ✎ levels={top secret, secret}
- ✎ categories={army,navy}



Preventing Information Flow with BLP

- no read-up
- no write-down



Read-down rule (simple security property) User with label L1 can read document with label L2 only when **L1 dominates L2**

Write-up rule (star property) User with label L1 can write document with label L2 when **L1 is dominated by L2**



Classified Documents Quiz

Select the best answer:

Since an unclassified document contains no sensitive information, it can be read or written by anyone in a system that implements BLP

☐ True

☐ False

BLP allows an unclassified user to write a top secret document.

☐ True

☐ False

Example

- Given the categories A, B, and C and the security levels TOP SECRET, SECRET, CONFIDENTIAL, and UNCLASSIFIED (ordered from highest to lowest), specify what type of access (read, write, or both) is allowed in each of the following situations. Assume that discretionary access controls allow anyone access unless otherwise specified.
- Lan, cleared for (TOP SECRET, {B, C}), wants to access a document classified (SECRET, {A, C}).
 - An, cleared for (TOP SECRET, {A, B}), wants to access a document classified (CONFIDENTIAL, {A}).
 - Ha, cleared for (CONFIDENTIAL, {A}), wants to access a document classified (CONFIDENTIAL, {B}).
 - Le, cleared for (SECRET, {A,C}), wants to access a document classified (SECRET, {A,C}).
 - Bi, who has no clearances (and so works at the UNCLASSIFIED level), wants to access a document classified (CONFIDENTIAL, {B}).

08/10/2024

57

Other MAC Models

• Biba is dual of BLP

- Focuses on **integrity rather than confidentiality**
- Read-up and write-down rules

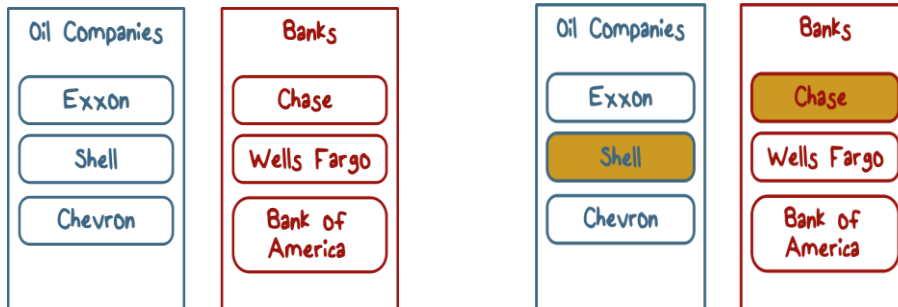


Example:

- Integrity level could be **high, medium or low**
- Compartment could be similar to BLP and captures topic(s) of document
- Low integrity information should never flow up into high integrity documents

Chinese Wall Policy (Conflict of Interest)

Objects are put into conflict classes:



The user **can access any object** as long as he/she has not accessed an object from another company **in the same conflict class**.

Trusted Computing Bases (TCB)

Revisiting Trusted Computing Base (TCB)



Trusting Software:

- How do we know TCB can be trusted?
- **Secure vs. trusted. vs high assurance**
 - Set of all hardware and software trusted to operate securely
 - Required for all other trust in the system security policy
- **Functional correctness**
 - Does what it was designed to do
- **Maintains data integrity**
 - Even for bad input
- **Protects disclosure of sensitive data**
 - Does not pass to untrusted software
- **Confidence**
 - Experts analyze program & assure trust
- **Statement giving security we expect system to enforce**
 - Do this formally when and where possible

TCB Design Principles

- **Least privilege for users & programs**
- **Economy**
 - Keep trusted code small as possible, easier to analyze & test
- **Open design**
 - Security by obscurity does not work
- **Complete mediation**
 - Every access checked, attempts to bypass must be prevented
- **Fail-safe defaults**
 - Default deny
- **Ease of use**
 - Users avoid security that gets in their way

How Do We Build a TCB:

Support Key Security Features

- **Must implement certain security relevant functions**
 - Authentication
 - Access control to files & general objects
 - Mandatory access control (SELinux)
 - Discretionary access control (standard file permissions)
- **Protection of data used by OS** (OS must protect itself)
 - Security features of trusted Oses
- **Complete mediation of accesses**
 - Trusted path from user to secure system
 - Prevents programs from spoofing interface of secure components
 - Prevents programs from tapping path (e.g. keyloggers)
- **Audit log showing object accesses** – only useful if you /look/ at the log
 - Detect unusual use of the system



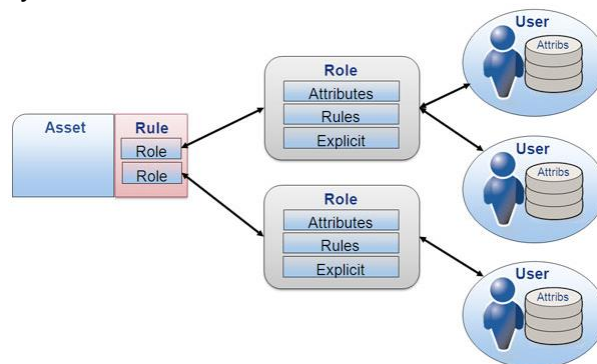
RBAC - role-based access control

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

RBAC Model – users, Roles, resources

∞ RBAC Model:

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



08/10/2024

66

Access Control Matrix Representation of RBAC

	R_1	R_2	$\bullet \bullet \bullet$	R_n
U_1	\times			
U_2	\times			
U_3		\times		\times
U_4				\times
U_5				\times
U_6				\times
\bullet				
\bullet				
\bullet				
U_m	\times			

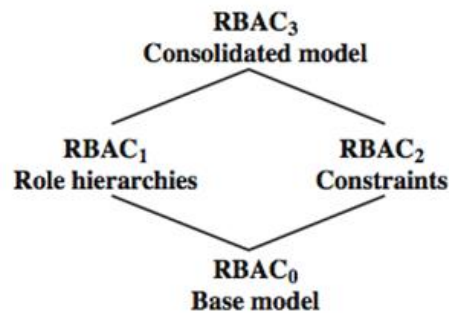
	OBJECTS								
	R_1	R_2	R_n	F_1	F_2	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R_2		control		write *	execute			owner	seek *
\bullet									
\bullet									
R_n			control		write	stop			

67

67

RBAC Model - Relationship among RBAC models

- $RBAC_0$: 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 system may be configured.
- $RBAC_3$: $RBAC_0$ + $RBAC_1$ + $RBAC_2$



(a) Relationship among RBAC models

- Constraints provide a means of adapting RBAC to the specifics of administrative and security policies in an organization. A constraint is a defined relationship among roles or a condition related to roles

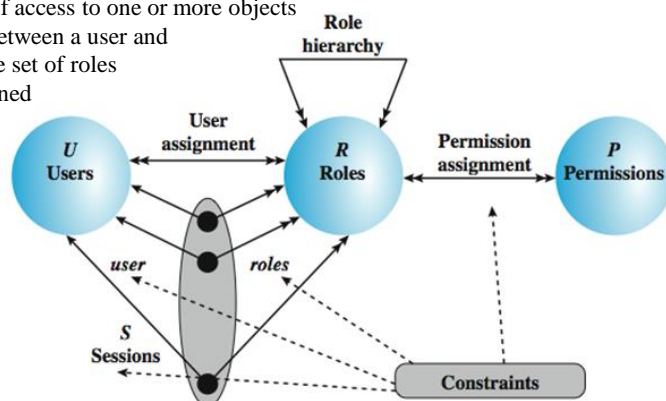
Hierarchies and Constraints

- ∞ 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

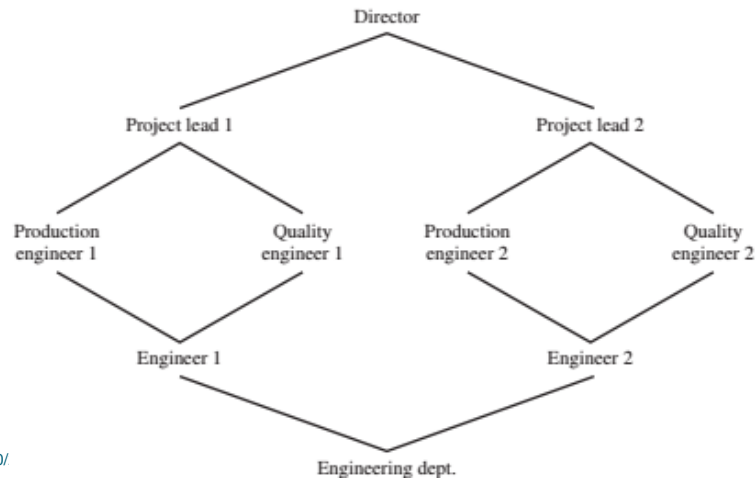
RBAC Model – an RBAC

- An RBAC system contains 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



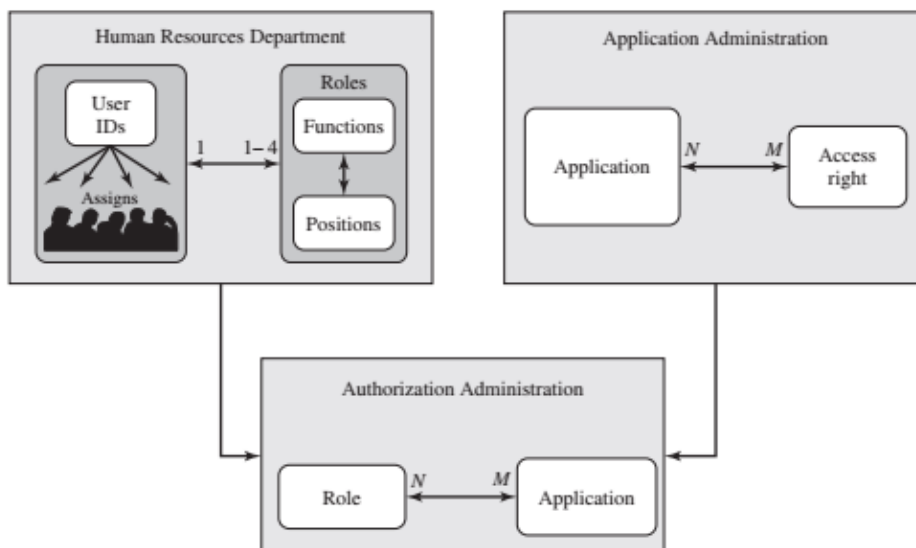
Example of Role Hierarchy



08/10/

71

Example of Access Control Administration



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.

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)

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)

ABAC - Attribute-based access control

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Attribute-based access control (ABAC)

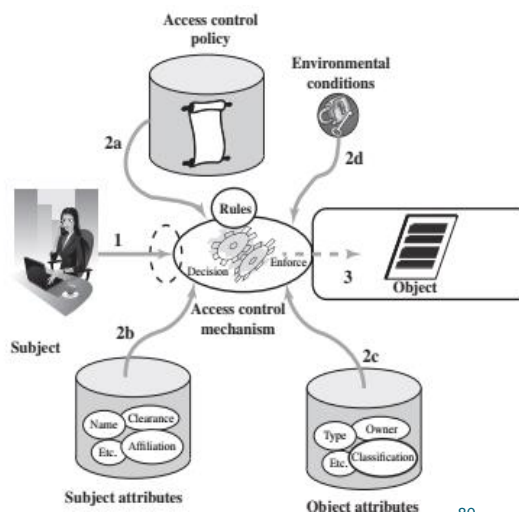
- There are three key elements to an ABAC model:
- attributes, which are defined for entities in a configuration;
 - a policy model, which defines the ABAC policies;
 - the architecture model, which applies to policies that enforce access control.

08/10/2024

79

ABAC Logical Architecture

- An access by a subject to an object proceeds according to the following steps:
- 1. A subject requests access to an object. This request is routed to an access control mechanism.
 - 2. The AC mechanism is governed by a set of rules:
 - (2a) that are defined by a preconfigured access control policy.
 - Based on these rules, the AC mechanism assesses the attributes of the subject (2b), object (2c), and current environmental conditions (2d) to determine authorization.

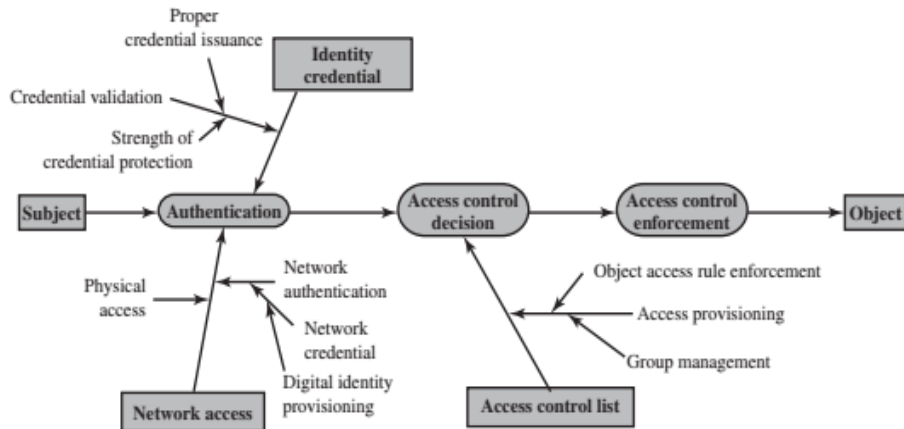


08/10/2024

80

ACL and ABAC Trust Relationships

ACL Trust Chain

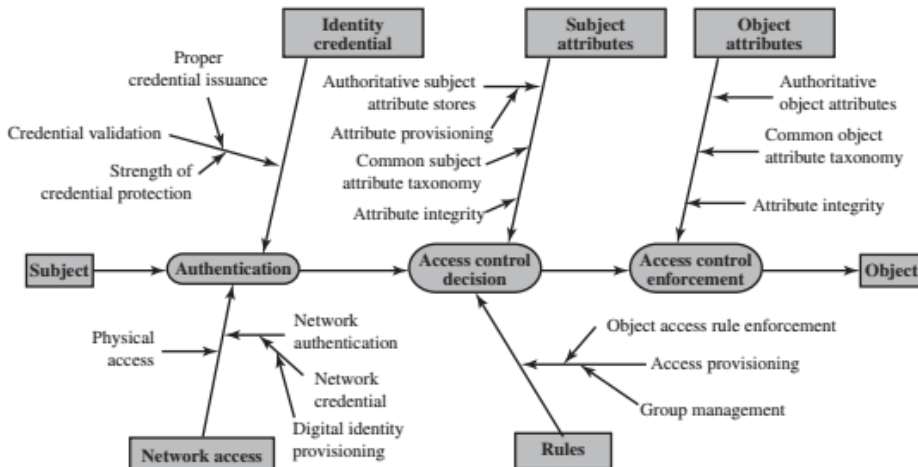


08/10/2024

81

ACL and ABAC Trust Relationships

ABAC Trust Chain



08/10/2024

82

ABAC Policies

- ✎ A **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.
- ✎ In turn, **privileges** represent the authorized behavior of a subject; they are defined by an authority and embodied in a policy.
- ✎ Other terms that are commonly used instead of privileges are **rights**, **authorizations**, and **entitlements**.

08/10/2024

83

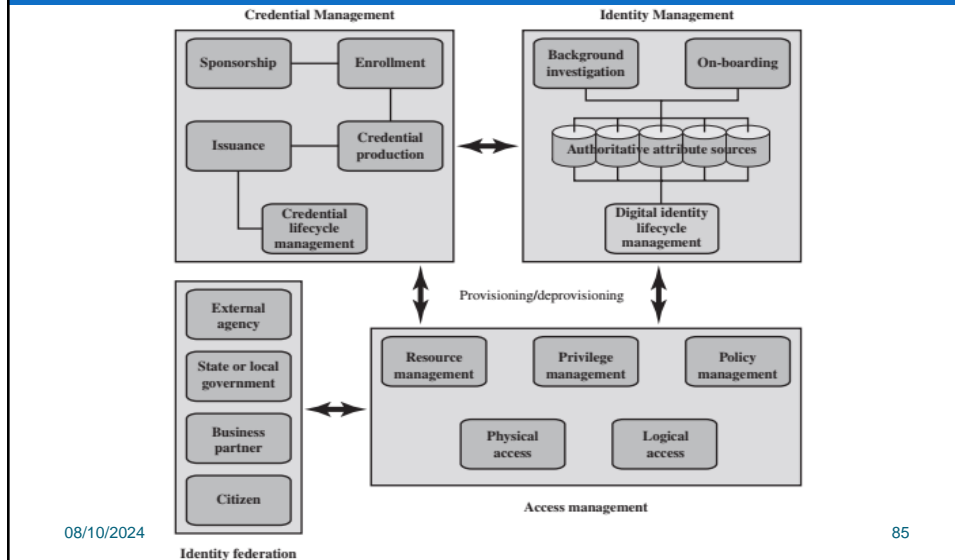
identity, credential, and access management (ICAM)

- ✎ ICAM is a comprehensive approach to managing and implementing digital identities (and associated attributes), credentials, and access control.
- ✎ ICAM: approach to access control, is designed to:
 - Create trusted digital identity representations of individuals and what the ICAM documents refer to as nonperson entities (NPEs).
 - Bind those identities to credentials that may serve as a proxy for the individual or NPE in access transactions.
 - A credential is an object binds an identity (and optionally, additional attributes) to a token possessed and controlled by a subscriber.
 - Use the credentials to provide authorized access to an agency's resources

08/10/2024

84

An overview of the logical components of an ICAM architecture



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.
- ✎ Special permission: permissions allow users and groups who are not the owner or group of a file to execute that file as though they were
 - SETUID - set user ID on execute
 - SETGID - set group ID on execute
 - StickyBit - puts the directory in sticky mode

Access Control Implementation in Unixlike Systems

Example: `chmod 4762 myfile` translates to:

```
setuid = on
setgid = off
sticky bit = off
user = read + write + execute
group = read + write
other = write
```

Set UID, GID, Sticky bit

```
chmod u+s = add setuid
chmod g-s = remove setgid
chmod o+t = add sticky bit
```

Others:

```
chmod a+w = add write to *all*
chmod a-wx = remove write and execute from *all*
chmod -R 755 myfolder
```

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 permissions
- Testing - quickly start a program as another user: **runas**

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

LAB

LABChapter 6