Computer and Network Security

Dr. Chan Yeob Yeun

Week 5 - 6

Weekly Lecture Plan

Wk	Contents	Cmt	Wk	Contents	Cmt
1	**Introduction**		9	Foundations of Network Security II	
2	**Foundations of Computer and Network Security**	**Tutorial Assig Plan**	10	Network-Based Threats and Attacks	
3	**Identification and Authentication I**		11	Network Security Protocols I	Quiz 2
4	**Identification and Authentication II**		12	Network Security Protocols II	
5	**Access Control**		13	Firewalls	
6	**Modern Computer Attacks**	**Quiz 1**	14	IDS / IPS	Assig Submit
7	Malicious Code	Assig Confirm	15	Revision and Presentation	
8	Foundations of Network Security I		16	Exam	

Outline

Access Control

Introduction

ACL

Bell-La Padula

Biba

Partial Ordering

Role-Based Access Control

Access Control consists of two steps:

Authentication

- Who or what is the subject (s)

Authorization

- Who or what is trusted to access object (o)

Access Control

We are going to look at several formal models for access control. Most of these models have the same structure:

The entities are divided into subjects and objects

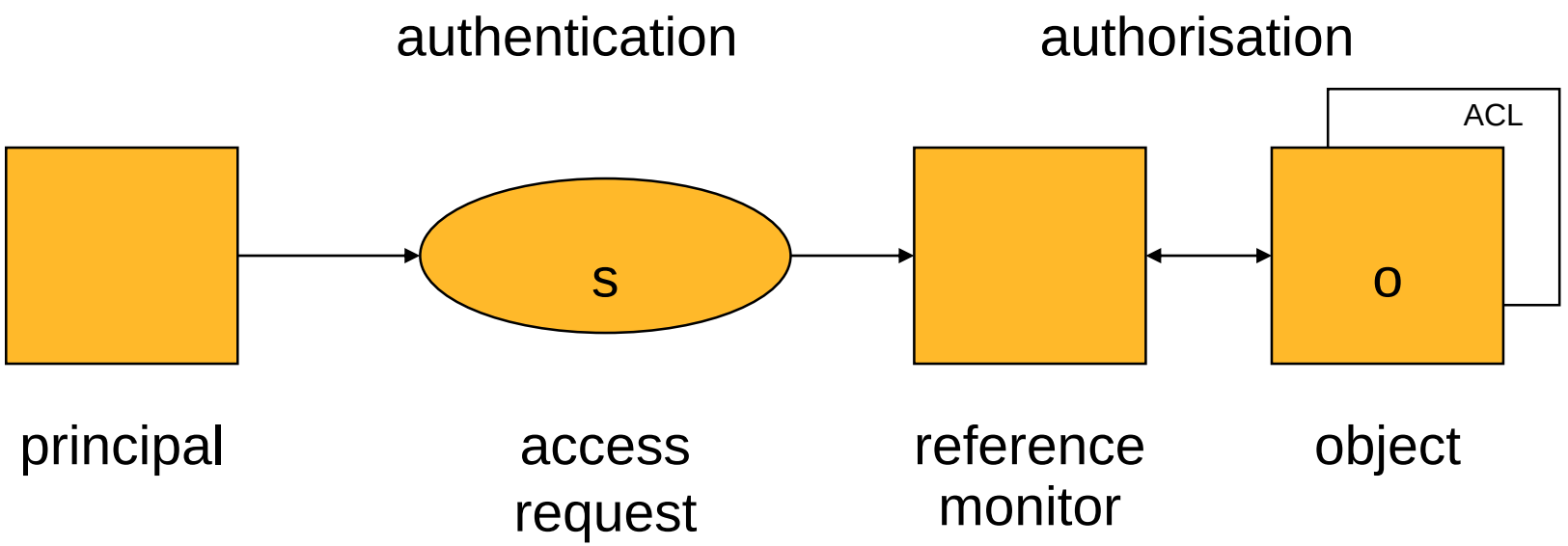
The notion of a “secure state” is defined

The permitted transitions from one state to another is defined by the access control policy

It is shown that these permissions move the system from a secure state to a secure state

Provided the system starts in a secure state, and the policies are enforced correctly, the system will remain in a secure state

Authentication & Access Control



B. Lampson, M. Abadi, M. Burrows, E. Wobber: Authentication in Distributed Systems: Theory and Practice, ACM Transactions on Computer Systems, 10(4), pages 265-310, 1992

Authorisation

An active entity, called ‘principal’ or ‘subject’, requests access to a passive entity, called ‘object’.

Authorisation: The reference monitor decides whether access is granted or denied.

The reference monitor has to find and evaluate the (automated) security policy relevant for the current request.

User identities are one of the parameters considered in security policies.

Subjects operate on behalf of human users we call principals, and access is based on the principal’s name bound to the subject in some unforgeable manner at authentication time.

Because access control structures identify principals, it is important that principal names be globally unique, human-readable and memorable, easily and reliably associated with known people. [M. Gasser, 1990]

Unix Access Rights

Three access operations:

- read: from a file
- write: to a file
- execute: a file

Access operations applied to a directory:

- read: list contents
- write: create or rename files in the directory
- execute: search directory

These operations differ from the Bell-LaPadula model. Unix write access does not imply read access.

Moral: Do not use your own intuition when interpreting access operations someone else has defined!

More Access Rights

Policies for creating and deleting files can be expressed by
- access controls on the directory (Unix)
- specific create and delete rights (Windows, OpenVMS)

Policies for defining security settings such as access rights could be handled similarly:
- access control on the directory
- specific rights like grant and revoke

Rights in CORBA: get, set, use, manage

Who Sets the Policy?

Security policies specify how principals are given access to objects. Two options for deciding who is in charge of setting the policy:

The owner of a resource decrees who is allowed access. Such policies are called discretionary as access control is at the owner's discretion.

A system wide policy decrees who is allowed access. Such policies are called mandatory.

Access Control Structures

Requirements on access control structures:

1. The access control structure should help to express your desired access control policy.
2. You should be able to check that your policy has been captured correctly.

Access rights can be defined individually for each combination of subject and object.

For large numbers of subjects and objects, such structures are cumbersome to manage. Intermediate levels of control are preferable.

Access Control Matrix

We specify for each combination of subject and object the operations that are permitted.

When all your users (principals) are known individually, you can express your policy in an access control matrix, with a row for each principal and a column for each object

	bill.doc	edit.exe	fun.com
Alice	-	{exec}	{exec,read}
Bob	{read,write}	{exec}	{exec,read,write}

Access Control Matrix continued

The access control matrix is

- an abstract concept,
- not very suitable for direct implementation,
- not very convenient for managing security.

How do you answer the question: Has your security policy been implemented correctly?

Bell-LaPadula (Orange Book): access control matrix defines mandatory access control (MAC).

Capabilities

Focus on the subject:

access rights are stored with the subject

capabilities ≡ rows of the access control matrix

Alice	edit.exe: {exec}	fun.com: {exec,read}

Subjects may grant rights to other subjects. Subjects may grant the right to grant rights.

How to check who may access a specific object?

How to revoke a capability?

Distributed system security has created renewed interest in capabilities.

Access Control Lists (ACLs)

Focus on the object:

access rights are stored with the object.

ACLs ≡ columns of the access control matrix.

fun.com	Alice: {exec}	Bill: {exec,read,write}

How to check access rights of a specific subject?

ACLs are implemented in most commercial operating systems but their actual use is limited.

Referring to individual users in a policy works best within organisations.

A management overhead has to be paid.

Groups

Alice and Bob are students in a large class; the lecturer wants to give students access to some documents;

Entering all names into several ACLs is tedious so the lecturer defines a group, declares the students to be members of the group, and puts the group into the ACLs

Access rights are often defined for groups:

- Unix: owner, group, others

Roles and RBAC

Alternatively, in our example we could have created a role ‘student’.

Definition: A role is a collection of procedures assigned to users; a user can have more than one role and more than one user can have the same role.

The lecturer would create a procedure for reading course material and assign this procedure to the role ‘student’.

A role ‘course tutor’ could be assigned a procedure for updating documents.

Procedures: ‘high level’ access operations with a more complex semantic than read or write; procedures can only be applied to objects of certain data types.

Example: funds transfer between bank accounts.

Access Control

Discretionary Access Control (DAC)

Within discretionary access control, some of the control remains at the discretion of the object's owner, or anyone else who is authorized to control access to the object. The owner can determine who should have access rights to an object and what those rights should be. Commercial environments typically use DAC to allow anyone in a group to have access to a file. Typically DAC access rights can be changed dynamically.

Mandatory Access Control (MAC)

Mandatory access control means that the access control policy decisions are made beyond the control of an individual owner of an object. A central authority determines what information can be accessible by whom, and the owner cannot change access rights. An example of MAC occurs in military security, where an individual data owner does not decide who has a top-secret clearance, nor can the owner change the classification of an object from top-secret to secret.

Access Control

Two methods are commonly used for applying mandatory access control:

- Rule-based access controls: This type of control further defines specific conditions for access to a requested object. Each object will have an associated Access Control List (ACL) that can express the conditions necessary for authorized access. When an object is requested, the system will lookup the corresponding ACL, and see if it is allowed. Examples of what the ACL can depend on include properties of the subject, the action requested or the context.
- Lattice-based access controls: These can be used for complex access control decisions involving multiple objects and/or subjects. A lattice model is a mathematical structure that defines greatest lower-bound and least upper-bound values for a pair of elements, such as a subject and an object. When deciding whether to grant access or not, the system will input the properties of the subject and object into the lattice and use explicit properties to decide.

Access Control Lists (ACL)

An ACL is a mechanism for expressing authorization rules separately for each object. Each object has one such list and the list shows all subjects who should have access and what kind of access they have been granted. The important question of who can modify ACLs will depend on whether the system is based on DAC or MAC.

As the name suggests, the permissions in an ACL are arranged in a list. When the system is querying the list, it will start at the top and search through until it finds a match (in terms of the subject and the operation requested). How partial matches are handled (i.e. if the subject requests multiple operations on the object and each rule only specifies one) will vary from system to system.

Access Rights - Bell-La Padula Model

The Bell-La Padula model is a formal description of the allowable paths of information flow within a secure system. The goal of the model is to identify allowable communication where it is important to maintain secrecy.

- Preventing the unauthorized disclosure of information
- Unauthorized alternation of information is secondary

The Bell-La Padula security model only allows a subject to access an object in two ways: *read* or *write*.

The security class of an object "*o*" is denoted by the function *C(o)*.

- The security class of a subject "s" is denoted by the function *C(s)*.

Typical security classes are Top-Secret, Secret, Confidential, Restricted and Unclassified.

D. Bell and L. La Padula, "*Secure Computing Systems: Mathematical Foundation and Model*", MITRE Report, MTR 2547 v2, 1973.

Bell-La Padula Model

Simple Security Property

"A subject *s* may have *read* access to an object *o* only if *C(o)* ≤ *C(s)*"

This says that the security class (clearance) of the subject receiving a piece of information must be at least as high as the class (classification) of the information.

*-Property (Star Property)

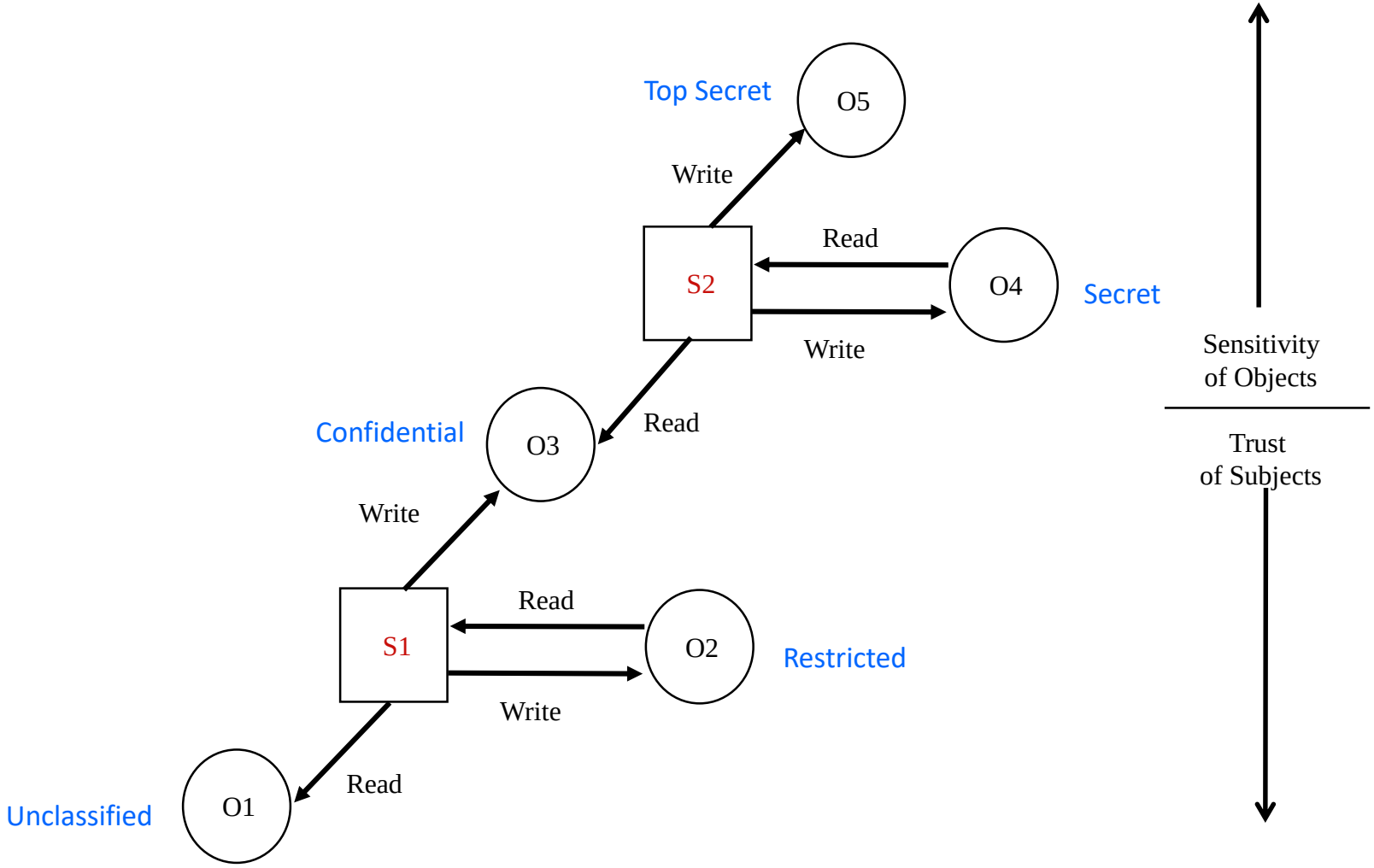
"A subject *s* who has *read* access to an object *o* may have *write* access to an object *p* only if *C(o)* ≤ *C(p)*"

This says that a subject obtaining information at one level may pass that information along only to other subjects at levels no lower that the information. This property is to prevent the Write-Down of information, which occurs when a subject with write access to high-level data transfers that data by writing it to a low-level object.

- Discretionary Security Property

"A subject s may access an object o if that right is contained in the discretionary access control matrix."

Implications of the Bell-La Padula Model



Access Control

The security models discussed have been focused primarily on confidentiality, in particular Bell-La Padula. Many commercial and industrial firms are more dependent on the accuracy of their data rather than preventing disclosure. We will now take a wider look at security models, and introduce a model for integrity policies.

The goals of any integrity model are:

1. Prevent unauthorized users from making modifications to data or programs
2. Prevent authorized users from making improper or unauthorized modifications to data or programs
3. Maintain the internal and external consistency of data and programs

Requirements for preserving data integrity

1. Users will not write their own programs, but will use existing production programs and databases.
2. Programmers will develop and test programs on a nonproduction system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
3. A special process must be followed to install a program from the development system onto the production system.
4. The special process in requirement 3 must be controlled and audited.
5. The managers and auditors must have access to both the system state and the system logs that are generated.

S. Lipner, “Non-Discretionary Controls for Commercial Applications,” *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pp. 111-112 (May 1999)

Requirements for preserving data integrity

These requirements amount to:

Separation of duty

- Developer and auditor

Separation of function

- Development system and production system

Auditing

- System logs

The Biba Model

In 1977, Kenneth J. Biba described a security policy designed to preserve data integrity. It can be considered the opposite of Bell-La Padula. Biba uses basically the same notation as previous models.

S is the set of subjects, O is the set of objects. Instead of security levels, we have *integrity levels*. These refer to the trustworthiness of the objects. The levels are ordered, and the function i() returns the integrity level of a subject or object.

There are three actions that are considered,

a subject reading an object,
a subject writing to an object,
and a subject invoking (executing) another subject.

The Biba Model (cont.)

Integrity levels have a very different meaning that security levels. Data at a higher integrity level is more accurate and/or reliable than data at a lower level. The higher the level, the more confidence one has that a program using the data will execute correctly.

Integrity levels inhibit the **modification** of information, security levels limit the **flow** of information.

The Biba Model (cont.)

The three rules of the Biba model are:

1. $s \in S$ can read $o \in O$ if and only if $i(s) \leq i(o)$.
2. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.
3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.

The first rule prevents a subject reading any data at a lower integrity level. If this were to happen then the subject would be contaminated by having to rely on data less reliable than its level (its integrity level would no longer be accurate).

This contamination is equivalent to disclosure in Bell-La Padula, it is what we are trying to prevent.

The Biba Model (cont.)

2. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.

This rule prevents a subject from one level writing to a higher level. If a subject were to alter a more trusted object, it could implant incorrect or false data (because the subject is less trusted than the object).

3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.

The third rule allows a subject to execute another subject provided the second subject is not at a higher integrity level. This is to prevent an invoker corrupting the execution of a more trusted subject. Execution is treated the same way as writing (contrast with Bell-La Padula).

So no matter how many different read and write operations are performed by any number of subjects, by enforcing these three rules, it is **impossible** for data at a low integrity level to corrupt data at a higher level.

FreeBSD has implemented the Biba model in its kernel.

Partial Ordering

The Bell-La Padula and Biba model define classes/classifications (security/integrity) for subjects and objects. This can be further refined into *categories*. The levels follow a strict (total) order (UNCLASSIFIED, CONFIDENTIAL, SECRET, TOP SECRET). Categories are arranged in what is called a *Partial Ordering*.

Each object can be assigned multiple categories. These are descriptors for the type of information contained. There is no quantitative measure for comparing them, but we still have to determine which is higher or lower for Bell-La Padula (no reads up) and Biba (no reads down).

Partial Ordering

A partial ordering is any comparison (≤) that obeys certain properties:

1.a ≤ a (reflexivity)

2.a ≤ b and b ≤ a ⇒ a = b (Antisymmetry)

3.a ≤ b and b ≤ c ⇒ a ≤ c (Transitivity)

This comparison needs only be defined for certain pairs of elements (hence *partial*). Ordinarily, using less-than-or-equal over the numbers defines a total-ordering. If we were to define an ordering as follows:

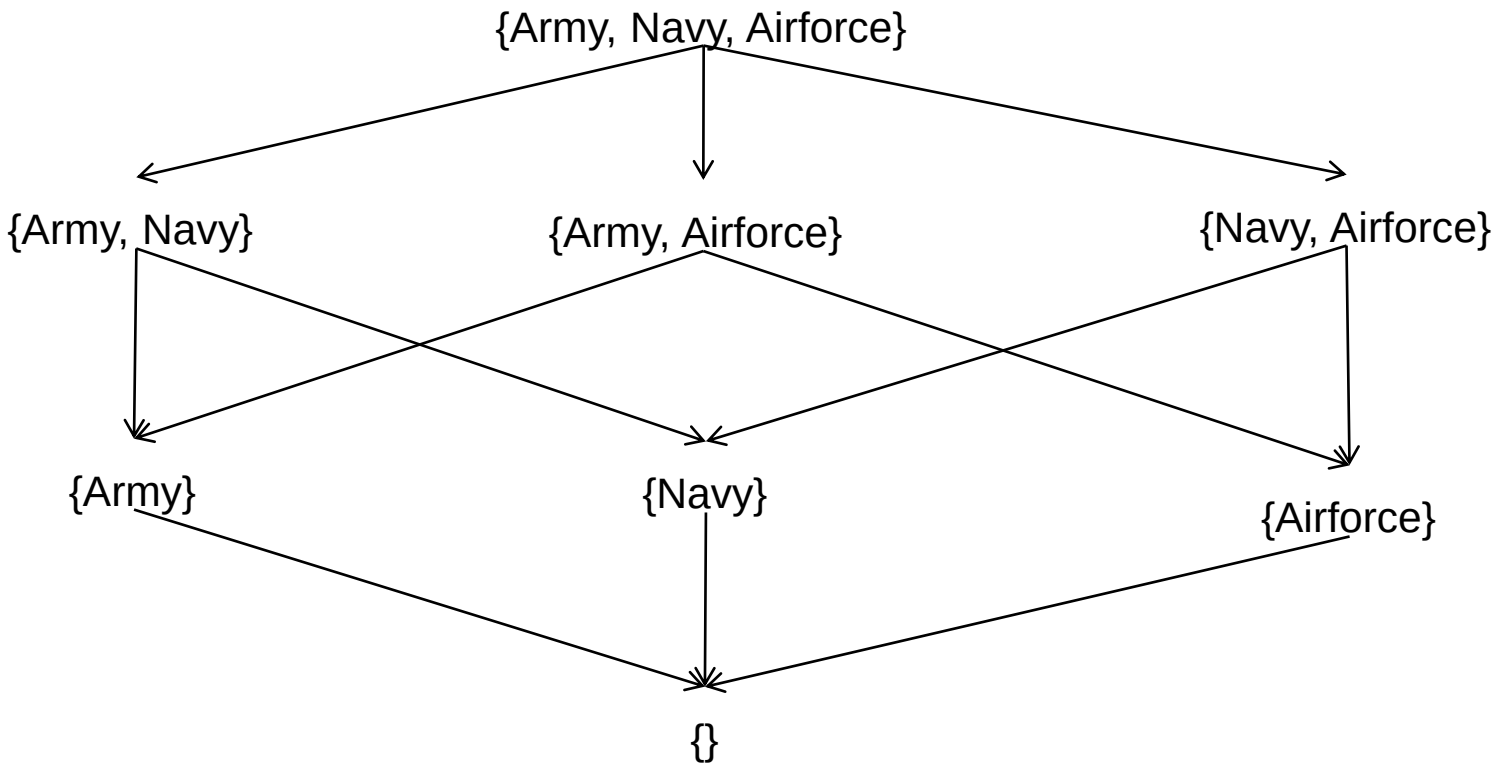
Scissors ≤ Rock, Paper ≤ Scissors, Rock ≤ Paper

this would not be a partial ordering, as it violates property 3.

When talking about compartments, we say c1 ≤ c2 if c1 ⊆ c2

For example: if c1 = {A,B}, c2 = {A,B,C} then c1 ≤ c2

Partial Ordering of categories



Arrows go from greater to lesser

Partial Ordering

Originally we defined Bell La-Padula in terms of security classes. Now we define it in terms of security levels, where the level is the combination of class and category:

E.g. Subject S might have (SECRET, {Army, Navy}), and Object O might have (TOP SECRET, {Army, Navy, Airforce})

The security level (L,C) *dominates* the security level (L',C') if and only if:

$$L' \leq L \quad \textit{and} \quad C' \subseteq C$$

The classification L' has to be less than L, and the categories C' must be contained in C. In BLP, the exact same rules remain ("no read up", "no write down"), but we now make all comparisons using both classes and categories.

Partial Ordering

There are two forms of *Tranquility*:

Strong Tranquility states that security levels do not change during the lifetime of the system

Weak Tranquility states that security levels do not change in a way that violates the rules of a given security policy.

An example of weak tranquility is using trusted entities that remove all sensitive information from a highly classified objected before its classification is changed to a lower one.

Examples

Consider a Bell-La Padula implementation with the following security classifications: TOP SECRET, SECRET, CONFIDENTIAL, UNCLASSIFIED, and the following categories: Army, Navy, Airforce. Who has what access to which documents of the below, and what kind?

Name	Security Level	Document	Security Level
Alan	SECRET, {Army, Navy}	Doc1	CONFIDENTIAL, {Army}
Brian	SECRET, {Army, Navy, Airforce}	Doc2	SECRET, {Navy, Airforce}
Clive	CONFIDENTIAL, {Navy}	Doc3	SECRET, {Navy}
Dan	TOP SECRET, {Army, Navy, Airforce}	Doc4	UNCLASSIFIED, {}

Examples

Consider a Biba implementation with the following integrity levels: CRUCIAL, VERY IMPORTANT, IMPORTANT, TRIVIAL and the following categories: Abu Dhabi, Dubai and Sharjah.

Name	Integrity Level	Document	Integrity Level
Alan	CRUCIAL, {Abu Dhabi}	Doc1	IMPORTANT, {Abu Dhabi}
Brian	VERY IMPORTANT {Abu Dhabi, Sharjah}	Doc2	VERY IMPORTANT, {Dubai}
Clive	IMPORTANT, {Abu Dhabi, Dubai}	Doc3	CRUCIAL, {Sharjah}
Dan	CRUCIAL, {Dubai, Sharjah}	Doc4	VERY IMPORTANT, {Dubai, Sharjah}

Role Based Access Control

Motivation

Mandatory Access Control can be quite restrictive. Access control models such as Bell La-Padula are only applicable in situations where confidentiality is of the utmost concern. Most commercial environments will have different priorities, for example integrity of information.

Discretionary Access Control allows users to grant or revoke rights for objects under their control, without the interference of an administrator. This does not always reflect the desired security policy.

It is more often the case that individuals do not own resources, but need access to them according to their job responsibilities. The organization is the “owner” of the resources.

Fortunately, there is another model that retains the central control of a single administrator, but is flexible enough for commercial use.

RBAC

In role-based access control (RBAC), the permissions to perform certain operations are assigned to specific roles.

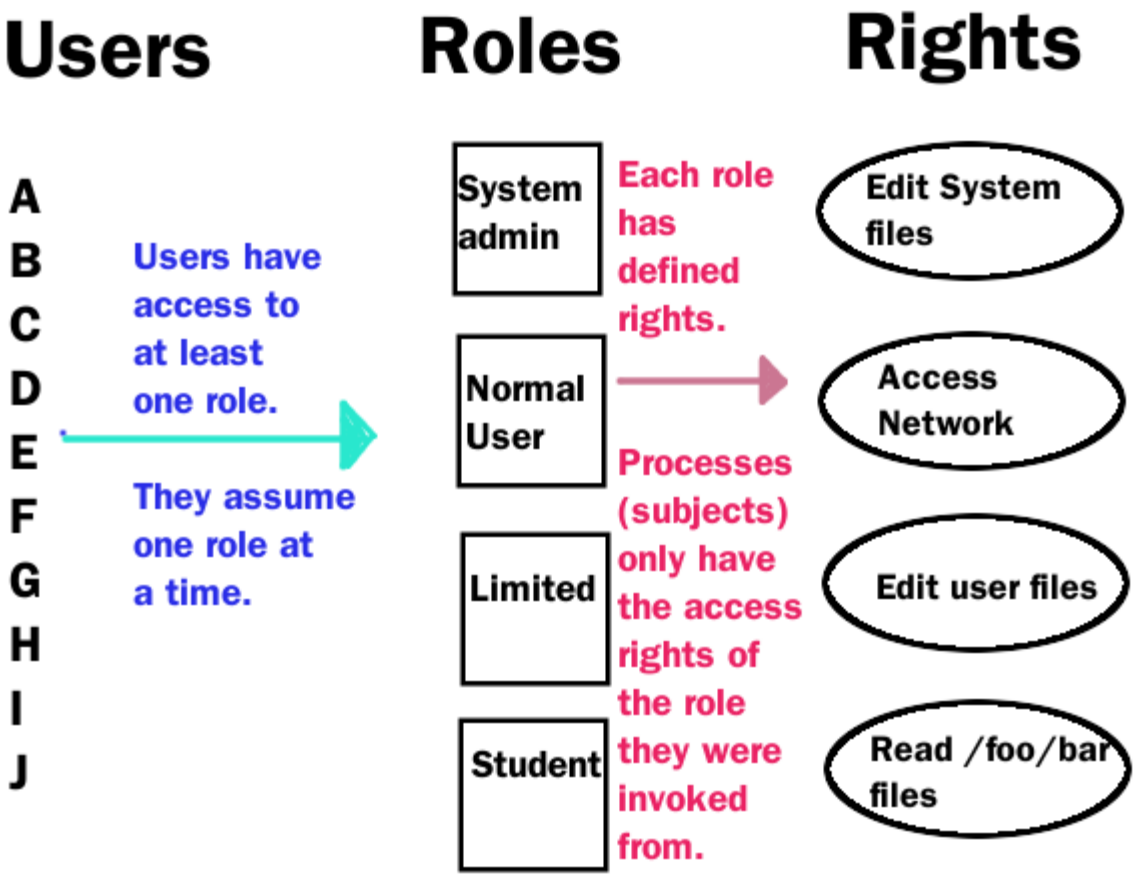
Members of staff (or other system users) are assigned particular roles

- Through those role assignments, users acquire the permissions to perform particular system functions.

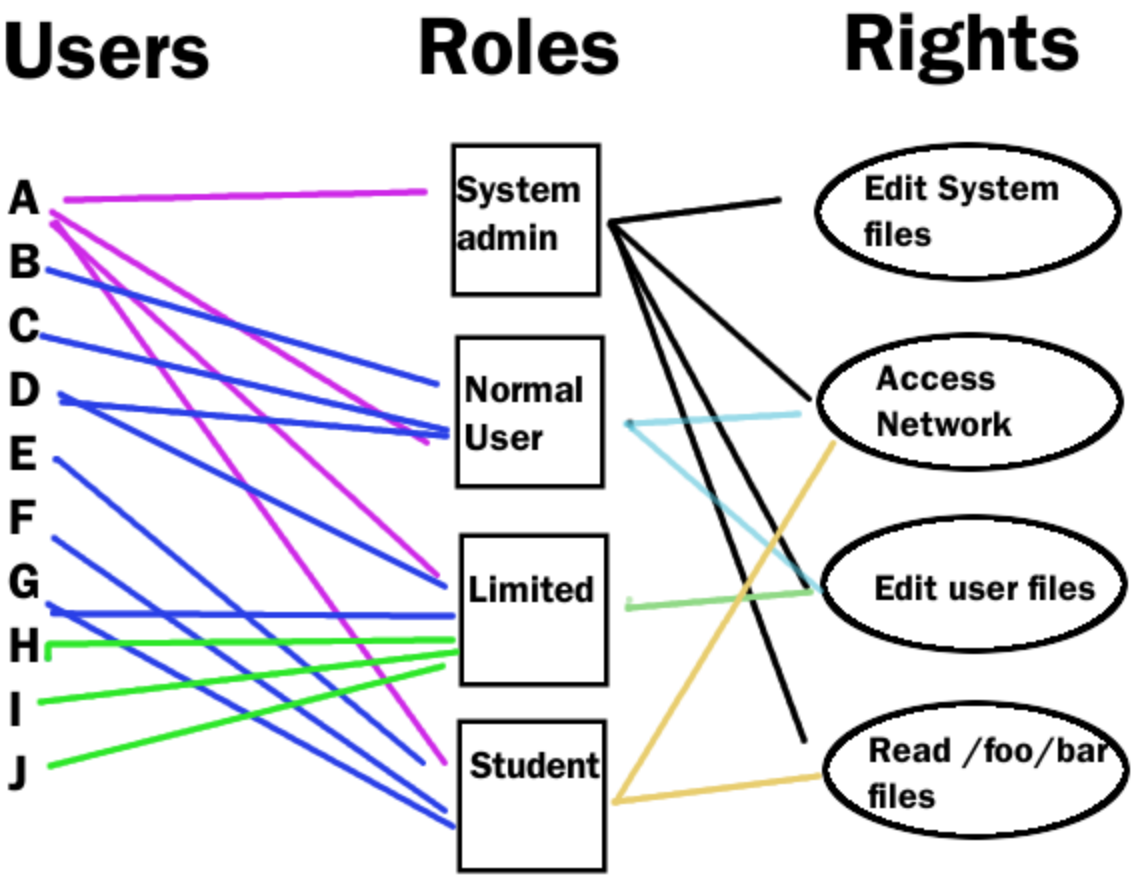
Since users are not assigned permissions directly, but only acquire them through their role (or roles), management of individual user rights becomes a matter of simply assigning appropriate roles to the user

- this simplifies common operations, such as adding a user, or changing a user's department.

RBAC Model


Users
Roles
Rights
A
B
C
D
E
F
G
H
I
J
Users have access to at least one role.
They assume one role at a time.
System admin
Normal User
Limited
Student
Each role has defined rights.
Processes (subjects) only have the access rights of the role they were invoked from.
Edit System files
Access Network
Edit user files
Read /foo/bar files

Using RBAC to manage access



MAC vs. RBAC

In RBAC, you don't need to modify and manage access control lists (ACLs), as the case for MAC.

ACLs created several challenges

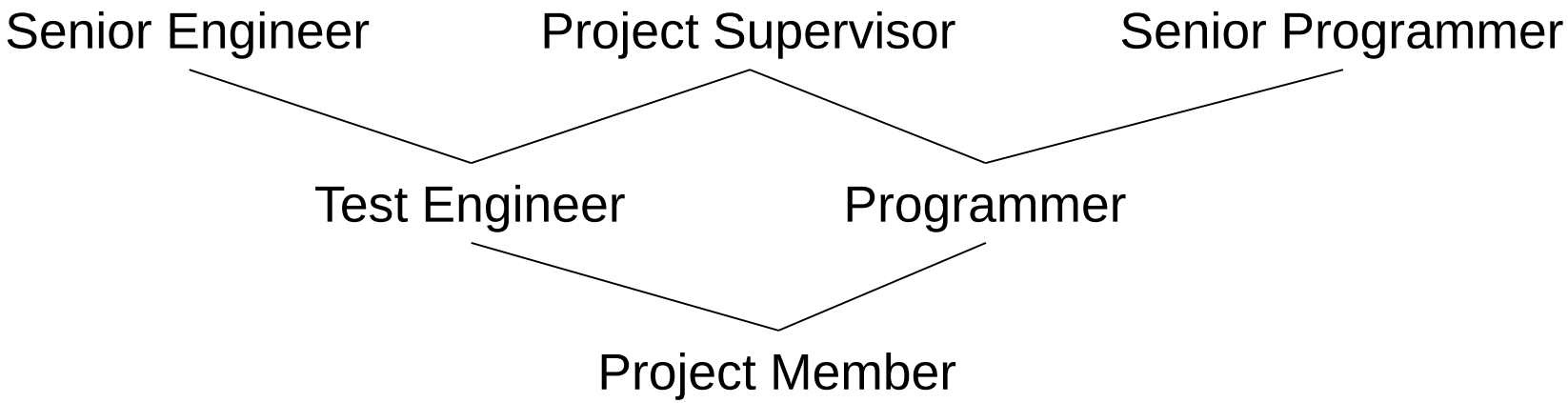
- modifying ACLs without causing unintended consequences,
- maintaining ACL modifications through upgrades, and
- troubleshooting problems that occurred due to using ACLs in a nonstandard way.

Role Based Access Control

Role Hierarchy

Roles can be arranged in hierarchies where one contains all the transactions of the other(s). This reduces redundancy where one role needs all the rights of another in addition to their own specific rights. This can be a simple way to delegate authority.

Note that containment of transactions might be the same way as the organizational hierarchy, or it could be inverted.



Role Based Access Control

Constraints

As well as prohibiting mutually exclusive roles, there are other constraints that may be desirable in RBAC. It might be necessary to limit:

Particularly sensitive permissions to a single role, to reduce the chances of misuse.

General membership in roles, e.g. a subject can be a programmer and a tester, but not on the same project.

The number of allowed subjects assigned to a role.

The number of allowed roles a subject may be assigned to/active.

Certain roles may have prerequisite roles.

Many of these are dependent on the correct mapping of subjects to real people. No protection can be assured if someone gains multiple IDs.

The Chinese Wall Model

The Chinese Wall model was proposed by Brewer and Nash in 1989[1], and its main aim is to prevent conflicts of interest.

Any organization that has different clients in the same market needs to be wary of conflicts of interest. If they have any sensitive information on one, that could be of competitive advantage to the other, then this information leaking is a risk (legal, financial, etc).

Consultancies, investment houses, legal firms all have to content with this problem, and all firms are required to take measures to prevent insider trading.

In essence:

There must be no information flow that causes a conflict of interest.

1. Brewer, Nash, “The Chinese Wall Security Policy”, IEEE Symposium on Research in Security and Privacy, 1989.