

Operating System Security

Özgür KAYA

Outline

- Motivation
- Core of Operating System : Kernel
- Kernel Components:
 1. Memory Management
 2. Process Management
 3. System Calls
- Secure OS vs. OS Security
- Why do we need Antivirus / Firewalls ?
- Securing the Gates
 - Information Flow
 - Secure Models
- Conclusion

Motivation - Step 1

1. Operating system is a software program *that enables the computer hardware to communicate and operate with the computer software*



Motivation - Step 2

2. Software has bugs that makes the program produce incorrect results, behave in an undesired way, or simply crash/terminate unexpectedly.



Motivation - Step 3

3. Bugs may lead to vulnerabilities *that can be exploited by attackers*



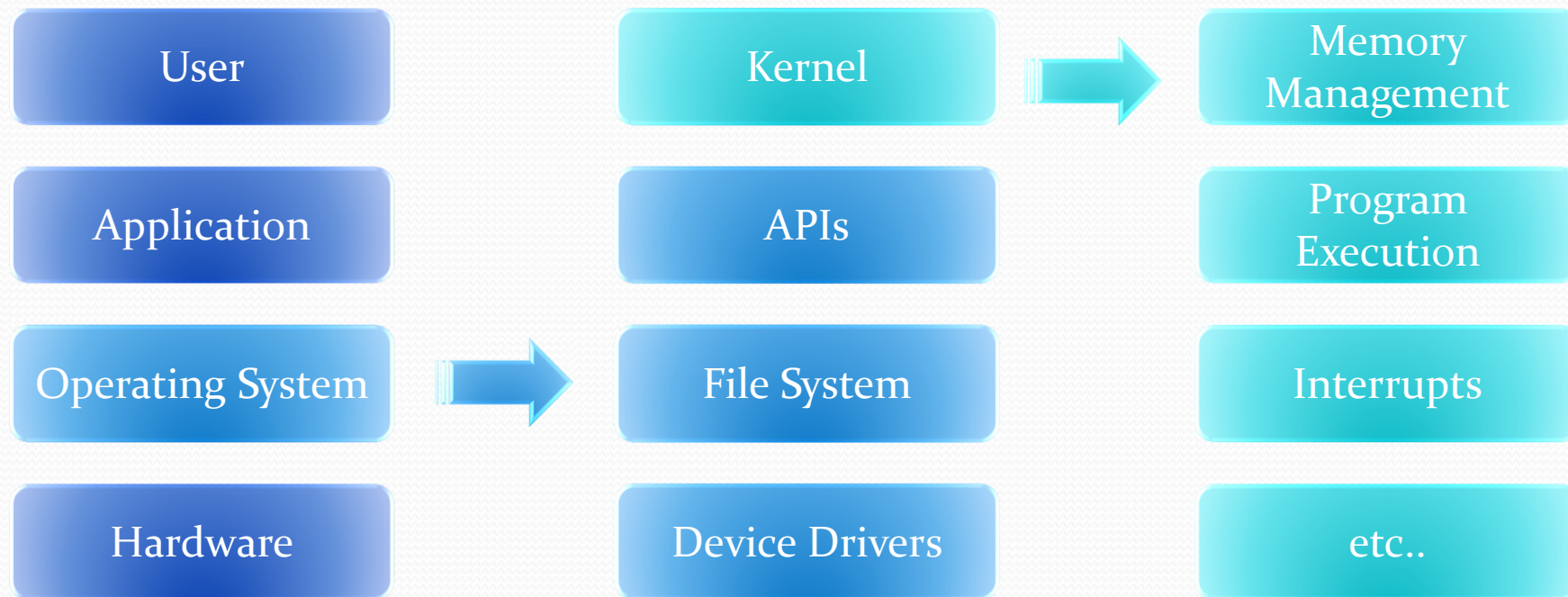
Motivation - Step 4

- Vulnerabilities should be defended *against attacks which may gain privillages by figuring out how to take advantages of them*

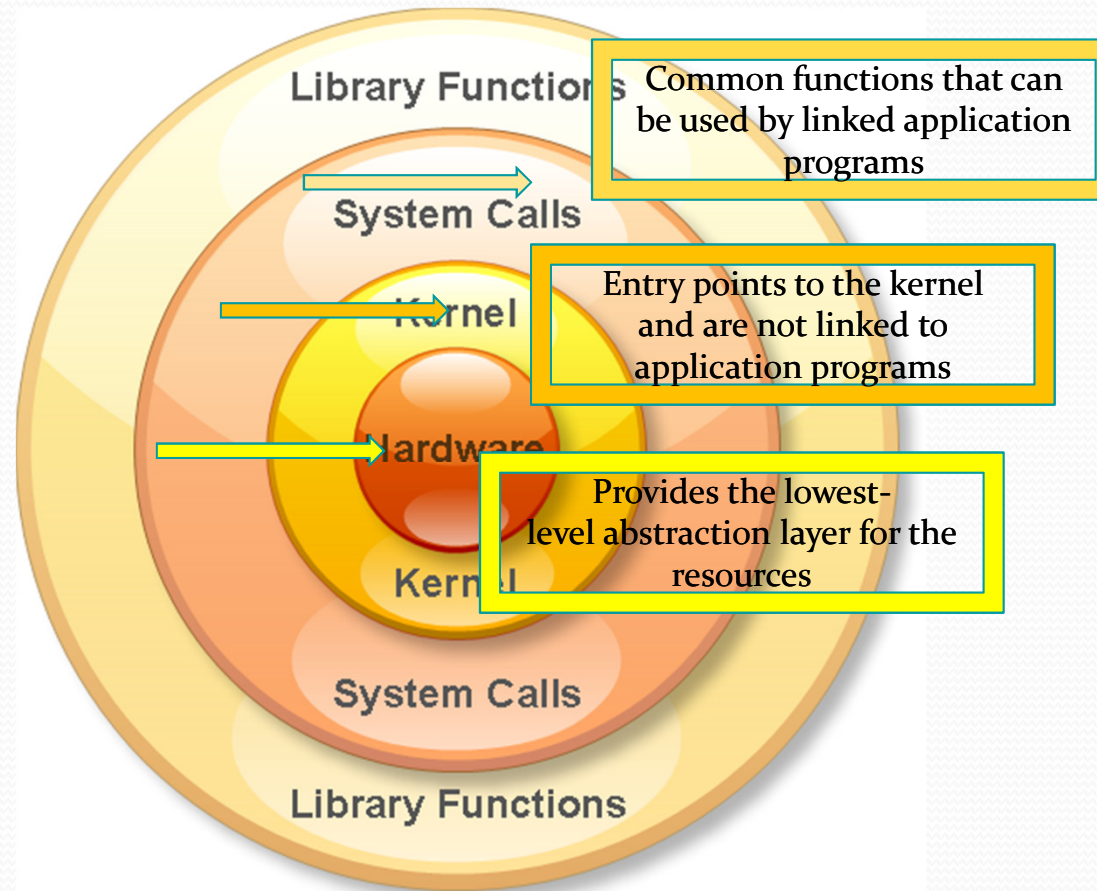


Core of Operating System: Kernel

- So... Where does security begin from ?



Kernel Overview

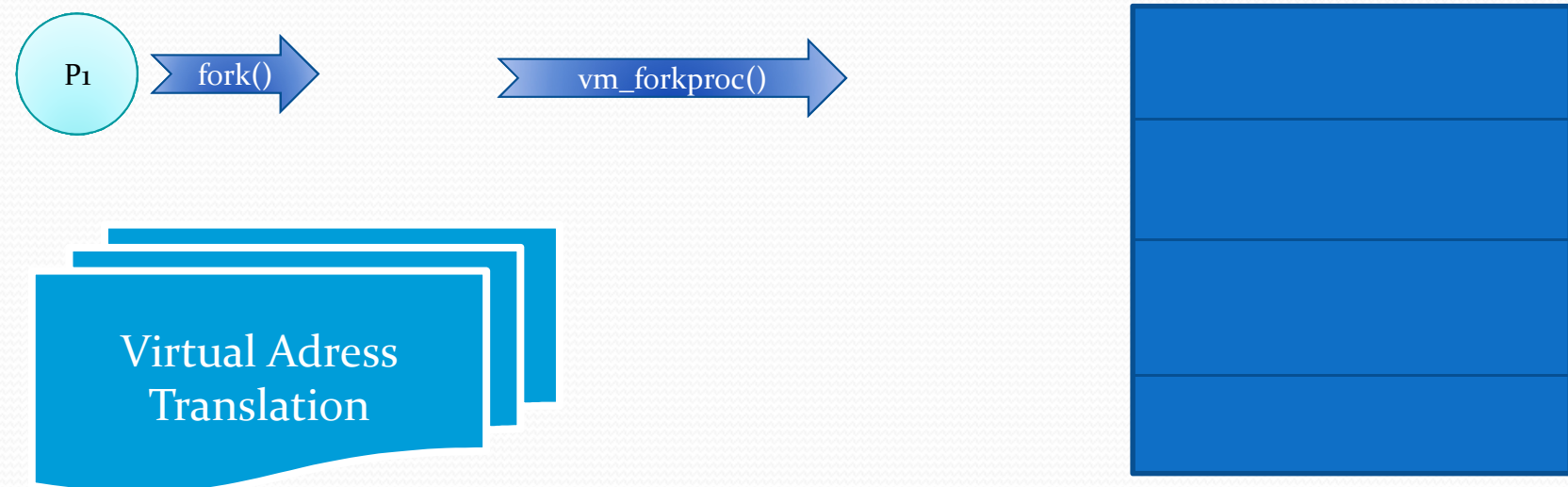


Kernel Components

- 1. Process management** *allows the execution of applications and support them with features such as hardware abstractions.*
 - **A process defines which memory portions the application can access.**
- 2. Memory management** *must perform processes to safely access this memory as they require it. Often the first step in doing this is virtual addressing, usually achieved by paging and/or segmentation.*
- 3. System calls** *is a mechanism that is used by the application program to request a service from the operating system.*
 - **It is impossible for a user process to call the kernel directly, because that would be a violation of the processor's access control rules.**

Example Scenario:

Running a program (in linux)



Virtual Address Translation

Page Number	Frame Address
2	101
3	102
4 (P_1)	101
5	103

Secure OS vs OS Security

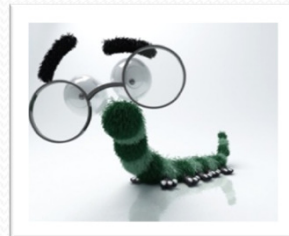
- Security Purposes

1. Authorization:



1. Is user X authorized to access resource R?

2. Authentication:



1. Who is the user?
2. Is the user really who he/she represents himself to be?

3. Integrity:



1. Ensuring that information is not altered by unauthorized persons

OS Security

- **Application Level Solutions:**
 - **Firewall** *is a software or hardware designed to permit or deny network transmissions based upon a set of rules*
 - **Antivirus** *software is used to prevent, detect, and remove malware, including computer viruses, worms, and trojan horses..*
 - **IDS** *is a device or software application that monitors network and/or system activities for malicious activities or policy violations and produces reports*

OS Security

- Defending something vulnerable..



Security Reports-1

- From a November 4 article by Gregg Keizer's on ComputerWorld:

Microsoft has been extremely busy patching pieces of the Windows kernel this year.

So far during 2011, Microsoft has patched 56 different kernel vulnerabilities with updates issued in February, April, June, July, August and October. In April alone, the company fixed 30 bugs, then quashed 15 more in July

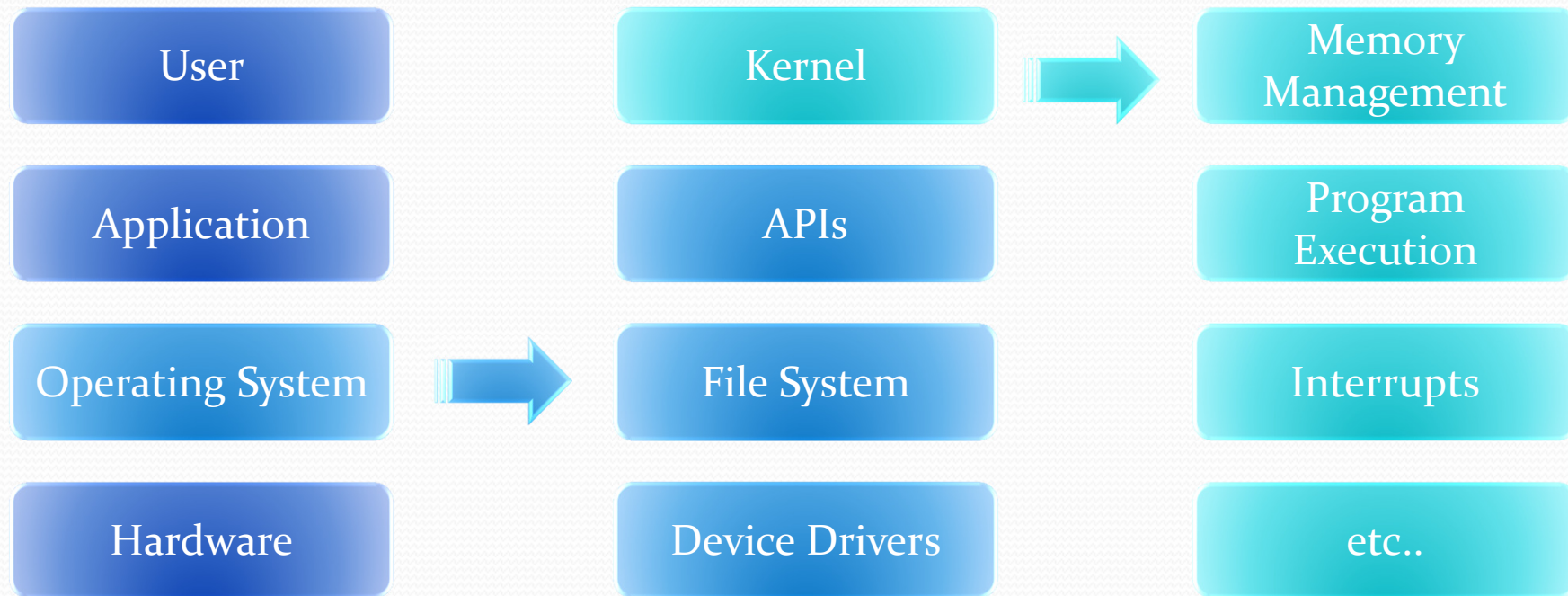
Security Reports-2

- Linux kernel vulnerabilities: State-of-the-art defenses and open problems (2011) :

141 Linux kernel vulnerabilities discovered from January 2010 to March 2011

Secure Operating Systems

- So... Where does security begin from ?



Securing the Gates: Access Control Models & Information Flow

- ACM -> *Deal with information flow*
- Information Flow -> *Confidentiality & Integrity (some extend)*

Information Control Model & Policies

- Concerns with the flow of information from one **security class** to another.
- Security Class: assigned to every object.
- Denning defined the concept of an Information flow policy as follows...

Denning's IFP

- Definition 1 [Information Flow Policy]

$$\langle SC, \rightarrow, \oplus \rangle$$

SC

set of security classes

$\rightarrow \subseteq SC \times SC$

flow relation (i.e., can-flow)

$\oplus: SC \times SC \rightarrow SC$

class-combining operator

Denning's IFP

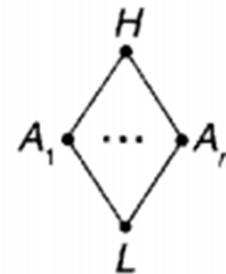
- All three components of Information Flow Policy are fixed.
- Allows objects to be created and destroyed dynamically
- However, Security classes can not.

Example 1

- Isolated Class:
 - No information flow is allowed from one SC to a different SC.
 - $SC = \{A_1..A_n\}$; for $i=1$ to n we have $A_i \rightarrow A_i$ and $A_i \oplus A_i = A_i$; and for $i,j = 1$ to n , $i \neq j$ we have $A_i \not\rightarrow A_j$ and $A_i \oplus A_j$ is undefined

Example 2

- High Low Policy:
 - All flows are allowed from Low to High
 - $SS = \{H,L\}$ and $\rightarrow \{(H,H), (L,L), (L,H)\}$ and join operation is;
 - $H \oplus H = H$, $L \oplus L = L$,
 $H \oplus L = H$, $L \oplus H = H$



Denning's IFP

- **Definition 2** [Denning's Axioms]
 $\langle SC, \rightarrow, \oplus \rangle$
 - 1 SC is finite
 - 2 \rightarrow is a partial order on SC (reflexive, transitive, antisymmetric)
 - 3 SC has a lower bound L such that $L \rightarrow A$ for all $A \in SC$
 - 4 \oplus is a least upper bound (lub) operator on SC

Justification for 1 and 2 is stronger than for 3 and 4. In practice we may therefore end up with a partially ordered set (poset) rather than a lattice.

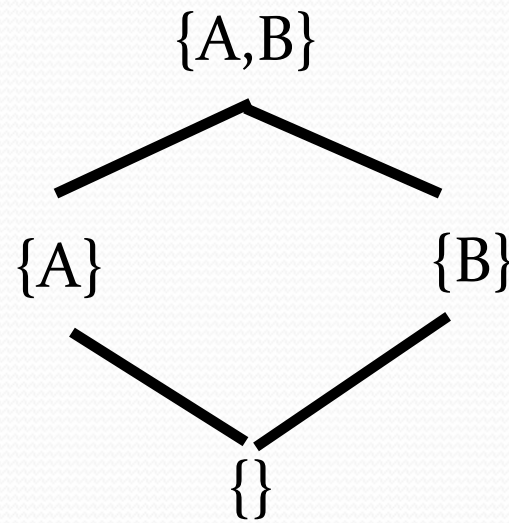
Example 3

- Bounded Isolated Classes
 - Example 1 fails by Axiom-4 ($A_i \oplus A_j$ is undefined)
 - $SC = \{A_1..A_n, L, H\}$; $L \rightarrow L$, $L \rightarrow H$, $H \rightarrow H$ than we can show $A_i \oplus A_j = H$

Denning's IFP

- Definition 3 [Dominance]
- The dominance relation has the following significance:
 - If $A > B$ then $A \not\rightarrow B$ but $B \rightarrow A$

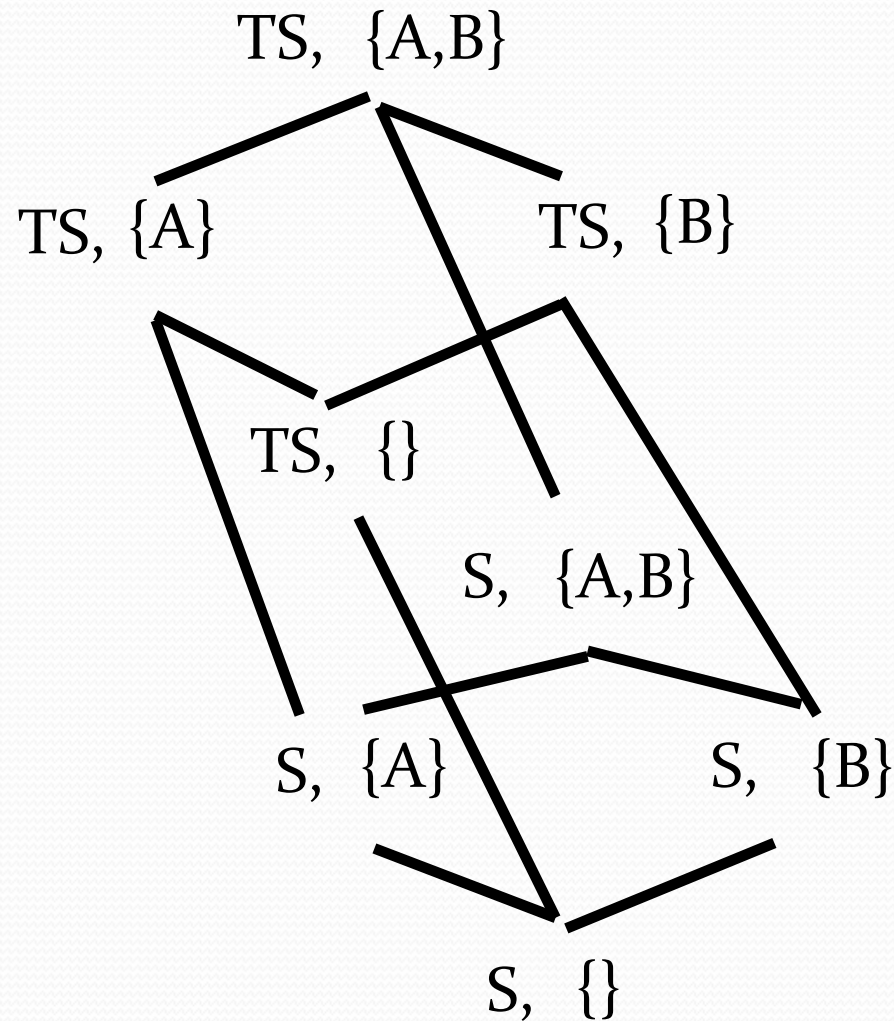
Lattice Structure



Hierarchical
Classes with
Compartments

product of 2 lattices is a lattice

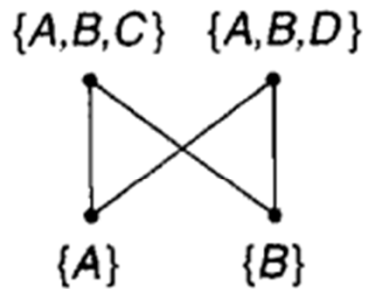
Lattice Structure



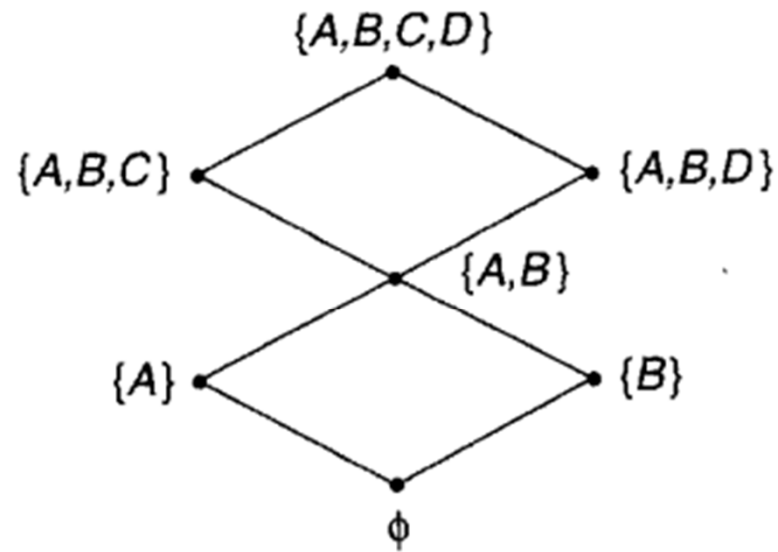
Hierarchical
Classes with
Compartments

Lattice Structure

- Partial Order to Total Order



(a)



(b)

Access Control Models

- Subject / User
 - Subject is a process that executes software behalf of User
 - Each User has one unique ID, Where each subject is associated with single user.
 - Each user can have many subject concurrently running
 - Different subject associated with the same user can obtain different sets of access rights

Access Control Models

- Assume TS user «John» logs as S
 - He can have subjects every level dominated by TS
 - Access rights presented by an [access matrix](#)
- A Subject can also be object
 - One process can execute/resume operations on another process

Access Control Models

- Discretionary Access Control:
 - Confidential user Tom wants Dick to read his file but not Harry
 - Tom enters access matrix: read in [Dick,File]
 - Than Dick also can state the same for Harry
 - What if Dick is not a cooperative ?
- Solution is to impose Mandatory Access Control

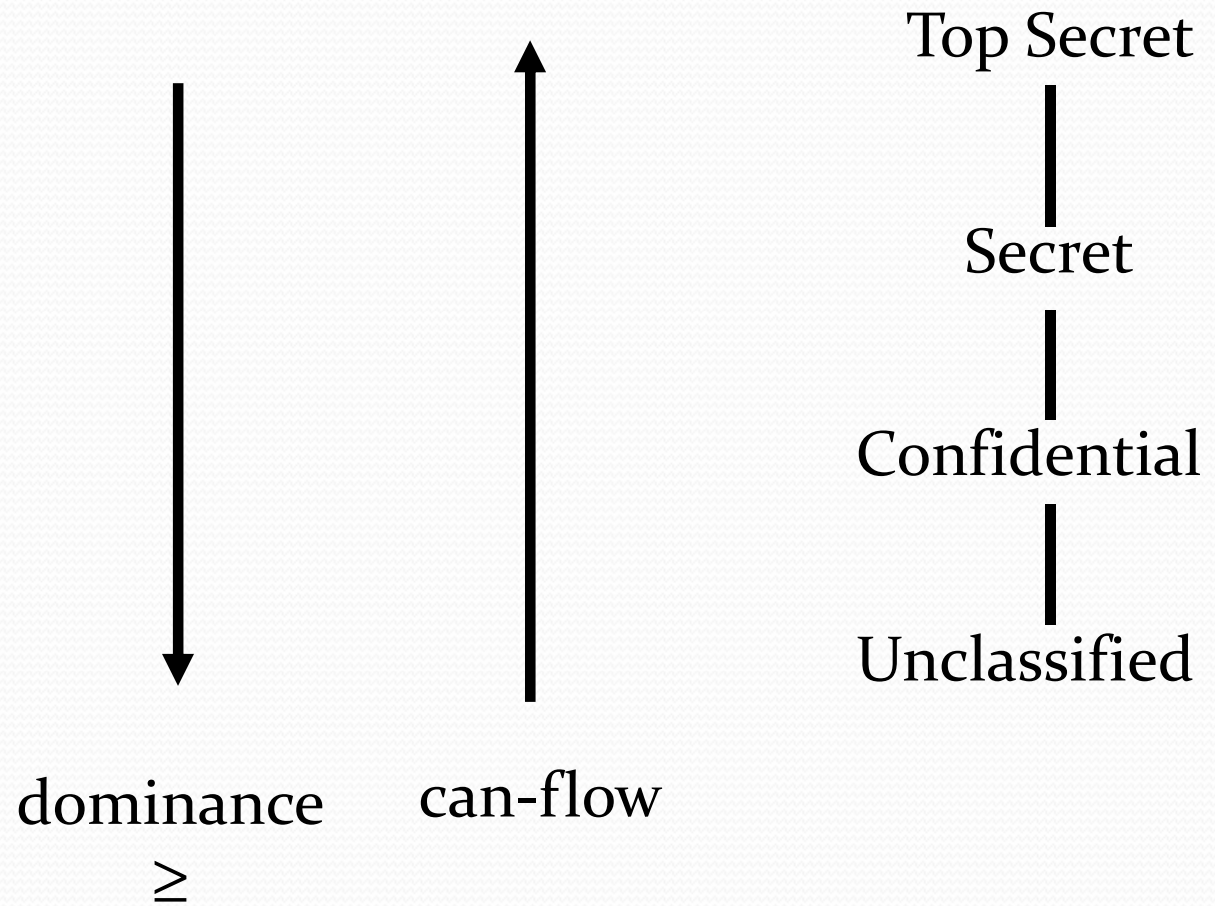
Bell-LaPadula Model

- The key idea in BLP is to augment DAC with MAC to enforce information flow policy
- 2 step approach:
 - DAM: modified by subjects
 - MAC: users have no control
 - Labels on subject: security clearance
 - Labels on object: security classification

Bell-LaPadula Model

- Same user can have multiple subject access same file with different previliges
- [Tranquility] The security labels on subjects and objects can not be changed

Bell-LaPadula Model



Bell-LaPadula Model

SIMPLE-SECURITY **No read up**

Subject S can read object O only if

- $\text{label}(S)$ dominates $\text{label}(O)$ (**TS can read S**)
- information can flow from $\text{label}(O)$ to $\text{label}(S)$

STAR-PROPERTY **No write down**

Subject S can write object O only if


- $\text{label}(O)$ dominates $\text{label}(S)$ (**S can write TS**)
- information can flow from $\text{label}(S)$ to $\text{label}(O)$

Bell-LaPadula Model

- SS implies Humans and Programs equally
- Star implies not Humans But Programs
- Ex: TS user can write S object ?
 - Star-Property prohibits
 - Need to log as S first
- No write down / No read up

Example

- Tom, Dick TS Users and have TS and S subjects
- Harry S User and only has S subjects.
- Tom create TS file with TS Subject
 - SS property prohibits Harry's Subject to read
- Even Dick has a Trojan Horse of Harry, Harry could not read the copy of file

- 
- Tranquility (most common): **SECURE**
label is static for subjects and objects
 - High water mark on subjects: **SECURE**
label is static for objects
label may increase but not decrease for subjects
 - High water mark on objects: **INSECURE**
label is static for subjects
label may increase but not decrease for objects

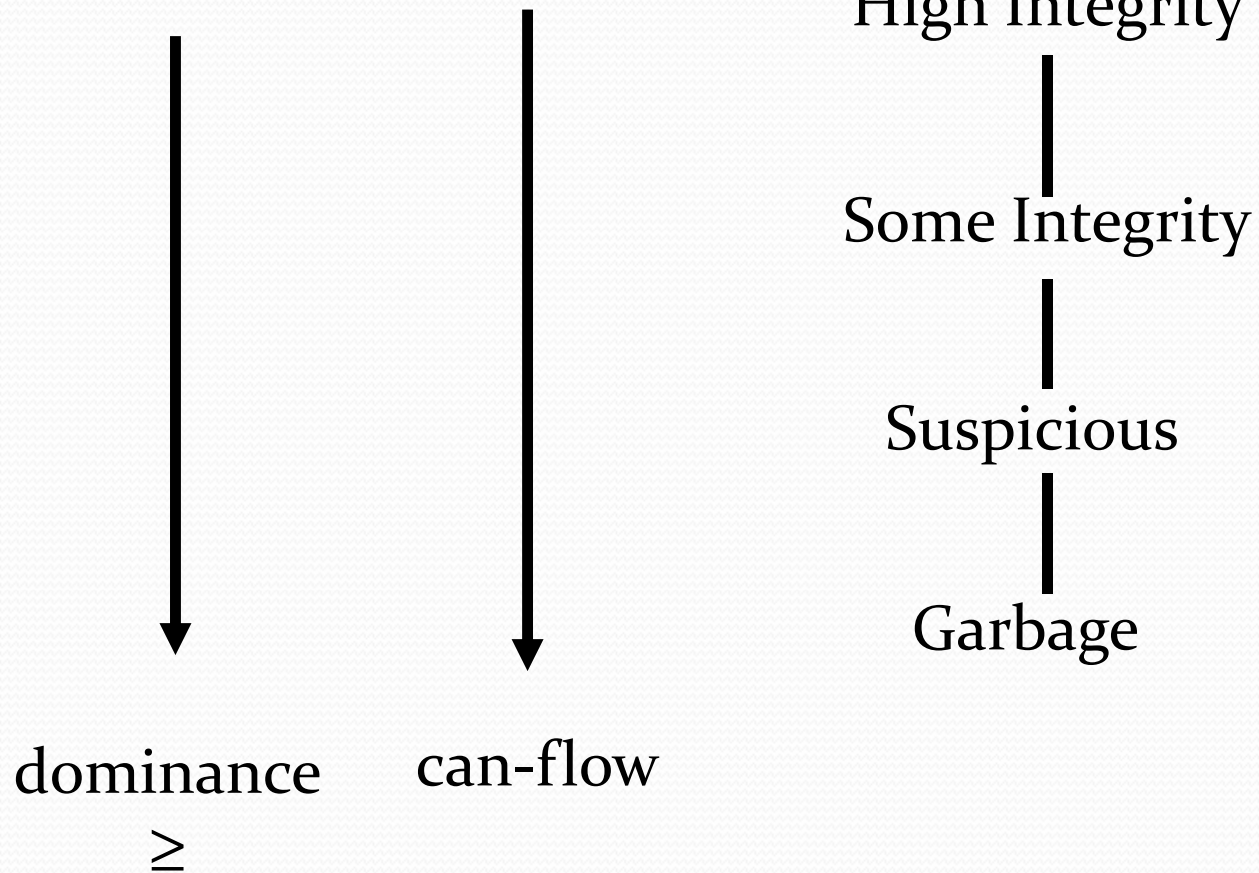
Bell-LaPadula Model

- Mandatory Access Control for Trojan Horse is enough ?
- TS can not write S But !
- Assume TS acquire large memory
- S can always request memory allocation for himself..
- Result with give a hint!
- Covert Channel Problem considered by Information Flow Models

Biba Model

- Biba proposed similar controls as BLP but for **Integrity**
- Information flows Top to Bottom
- There is no fundamental difference between Biba and BLP

Biba Model



Combining Biba and BLP

- Subject S can read object O only if $\text{conf}(s) \geq \text{conf}(o)$ and $\text{int}(s) \leq \text{int}(o)$
- Subject S can write object O only if $\text{conf}(s) \leq \text{conf}(o)$ and $\text{int}(s) \geq \text{int}(o)$
- By production of two lattices we can get a single lattice

Combining Biba and BLP

HS

HI



LS

LI

BLP

BIBA

GIVEN

\Rightarrow

HS, LI

HS, HI

LS, LI

LS, HI

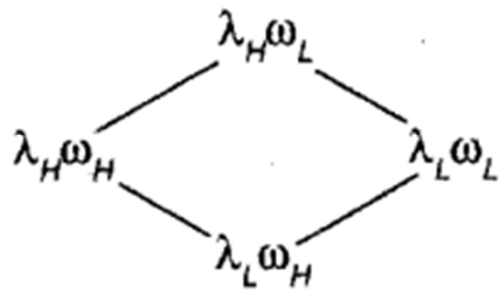
EQUIVALENT BLP LATTICE

$$\Lambda = \{\lambda_H, \lambda_L\}, \lambda_H \geq \lambda_L$$

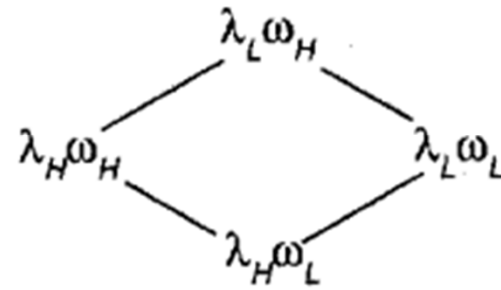
$$\Omega = \{\omega_H, \omega_L\}, \omega_H \geq \omega_L$$

	$\lambda_L \omega_L$	$\lambda_L \omega_H$	$\lambda_H \omega_L$	$\lambda_H \omega_H$
$\lambda_L \omega_L$	rW	r	W	ϕ
$\lambda_L \omega_H$	W	rW	W	W
$\lambda_H \omega_L$	r	r	rW	r
$\lambda_H \omega_H$	ϕ	r	W	rW

(a)

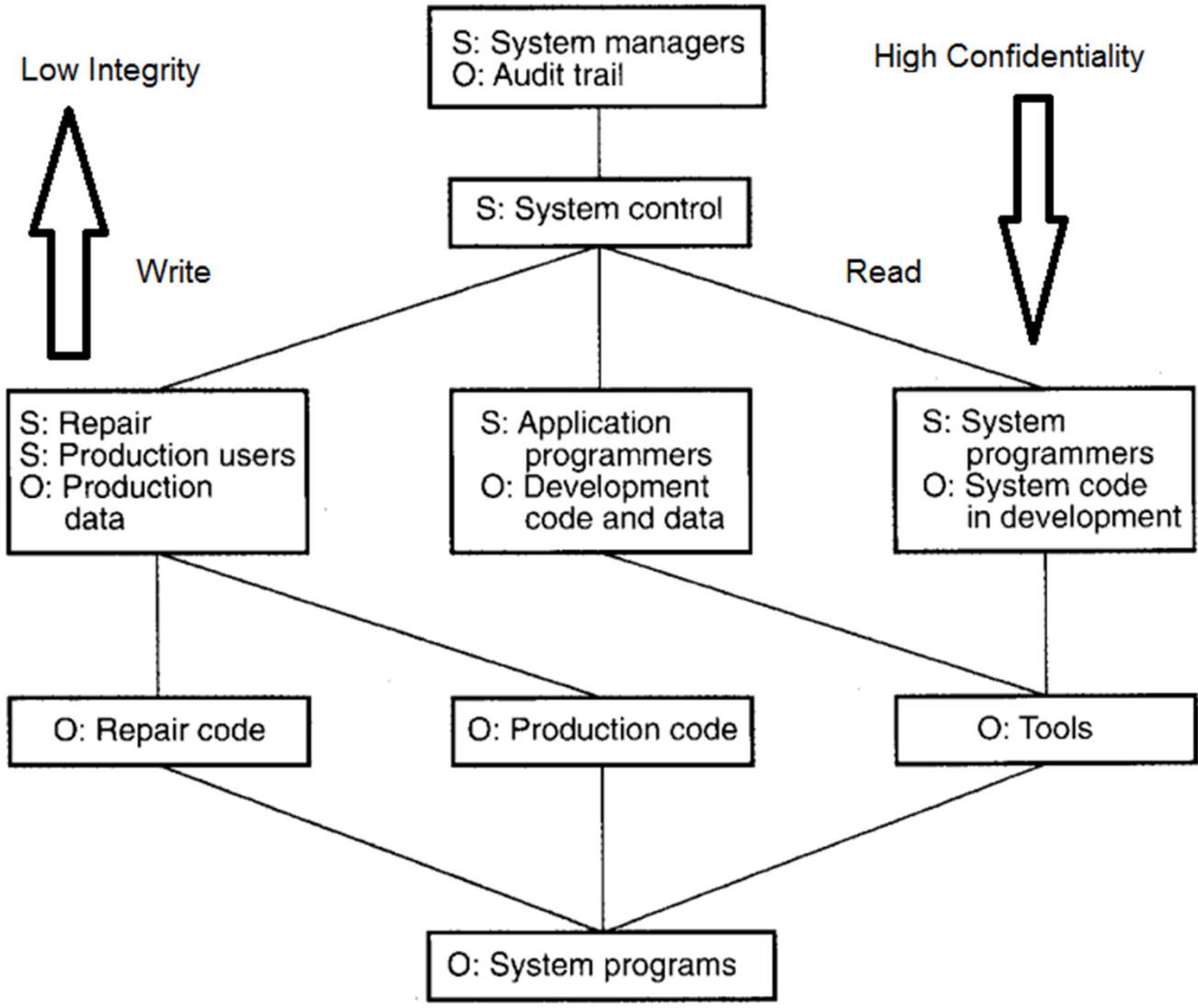


(b)



(c)

LIP



Chinese Wall Lattice

- Example of a commercial security policy for confidentiality
- Mixture of free choice (discretionary) and mandatory controls

Example

*CONFLICT OF INTEREST
CLASSES*

ALL OBJECTS

BANKS

OIL
COMPANIES

*COMPANY
DATASETS*

A

B

X

Y

A consultant can access information about at most one company in each conflict of interest class

Chinese Wall SIMPLE SECURITY

S can read O only if

- O is in the same company dataset as some object previously read by S (i.e., O is within the wall)

or

- O belongs to a conflict of interest class within which S has not read any object (i.e., O is in the open)

Chinese Wall STAR-PROPERTY

S can write O only if

- S can read O by the simple security rule

and

- no object can be read which is in a different company dataset to the one for which write access is requested

Star Property Example

ALICE'S WALL

Bank A

Oil Company X

BOB'S WALL

Bank B

Oil Company X

- ◎ cooperating Trojan Horses can transfer Bank A information to Bank B objects, and vice versa, using Oil Company X objects as intermediaries

Chinese Wall Conclusion

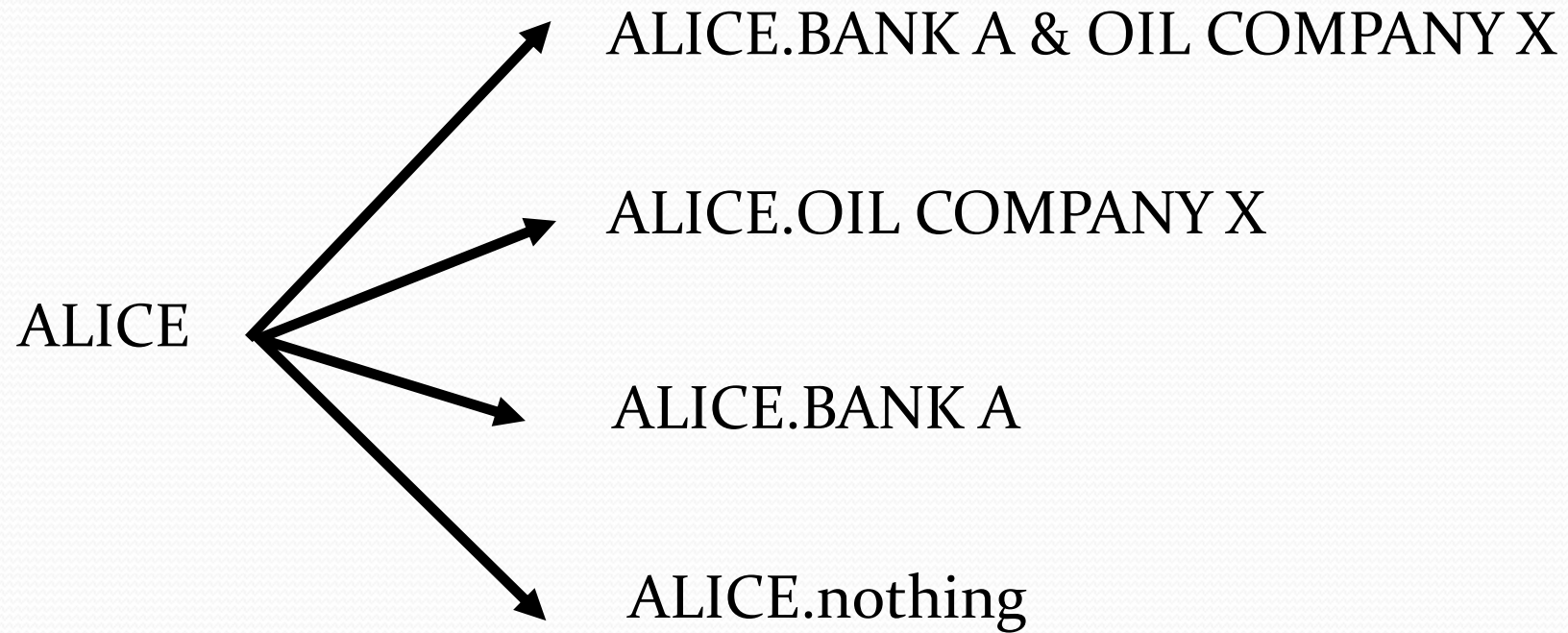
Either

- S cannot write at all

or

- S is limited to reading and writing one company dataset

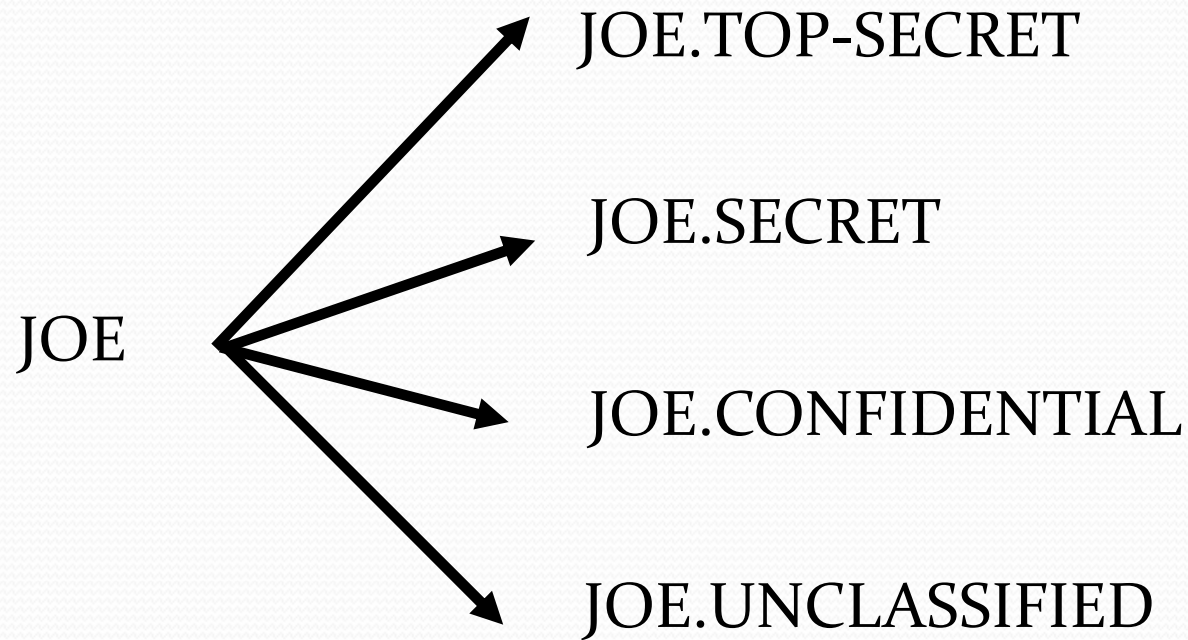
USERS, PRINCIPALS, SUBJECTS



USER

PRINCIPALS

USERS, PRINCIPALS, SUBJECTS



USER

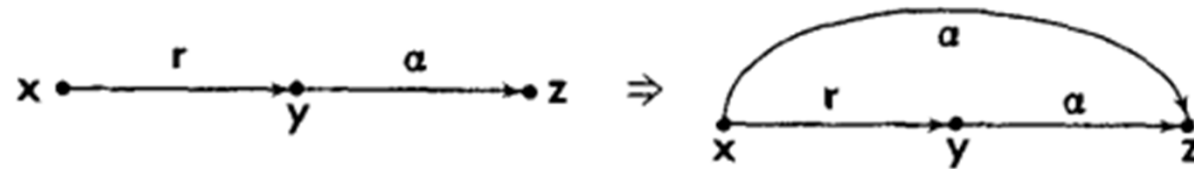
PRINCIPALS

- 
- The Bell-LaPadula star-property is applied not to Joe but rather to Joe's principals
 - Similarly, the Chinese Wall star-property applies not to Alice but to Alice's principals

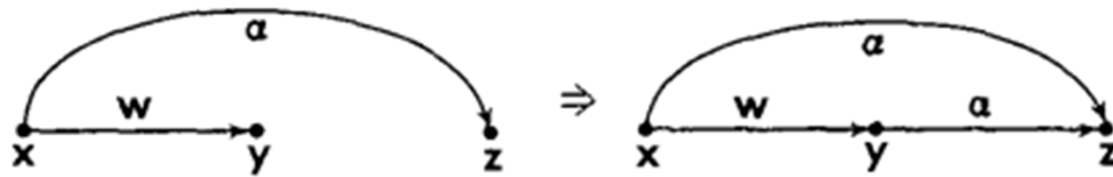
Take – Grant Model

- Present a concrete example of protection system
- Completely analyze its behaviour in linear time by using Graph approach

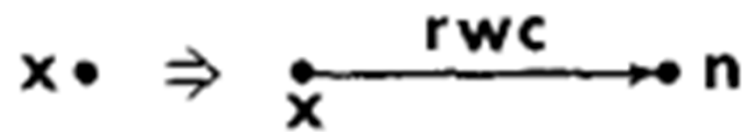
Rule 1. Take



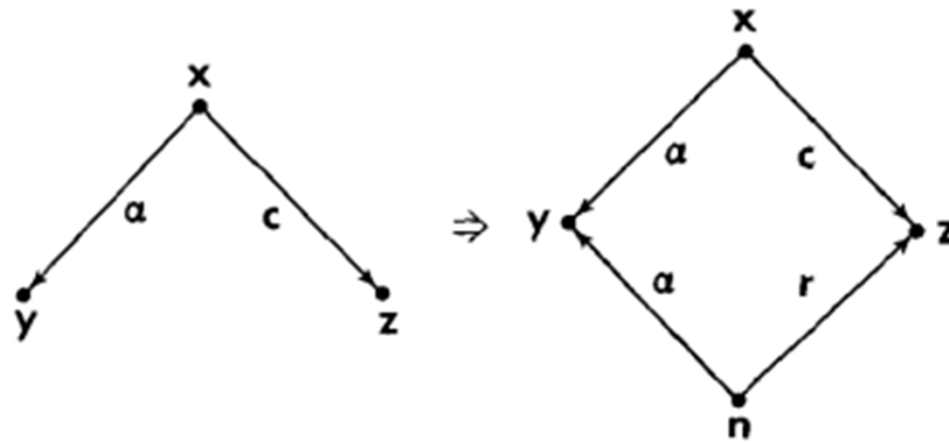
Rule 2. Grant



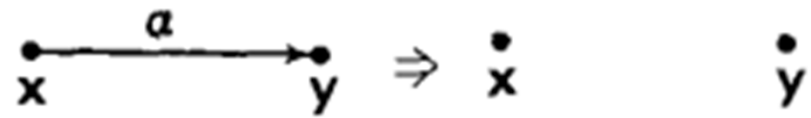
Rule 3. Create



Rule 4. Call

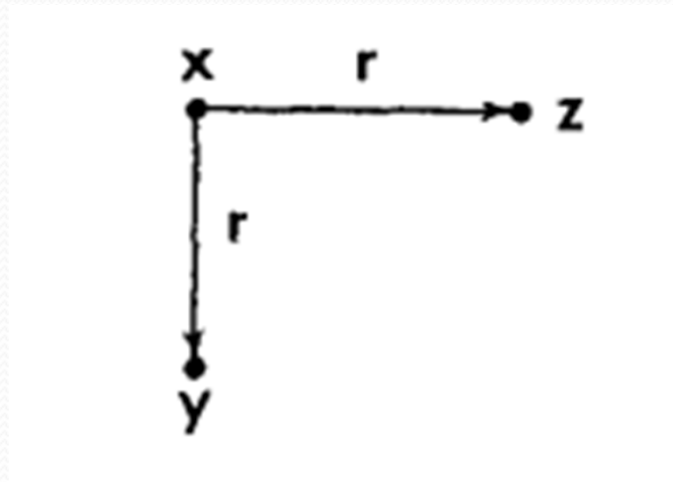


Rule 5. Remove

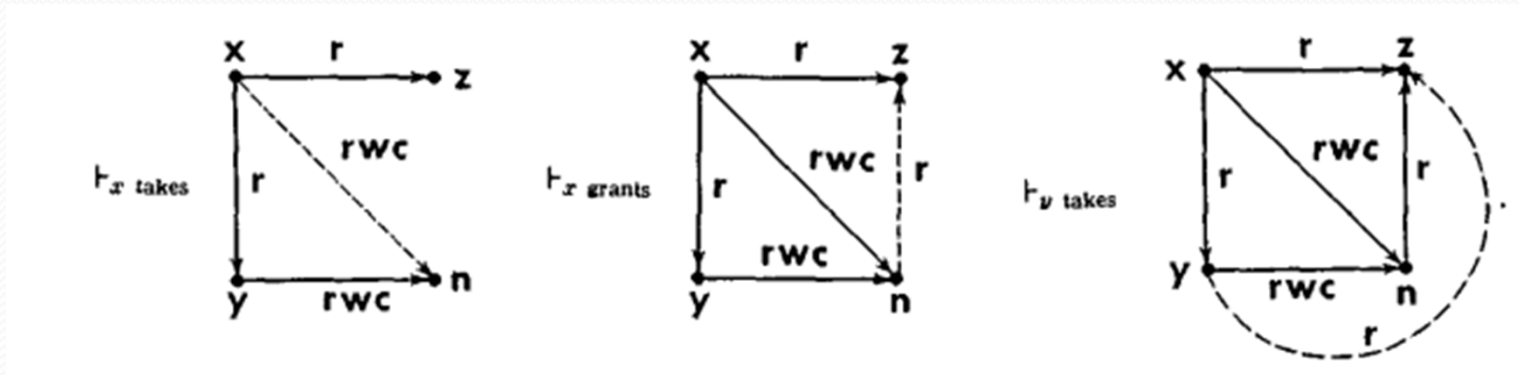
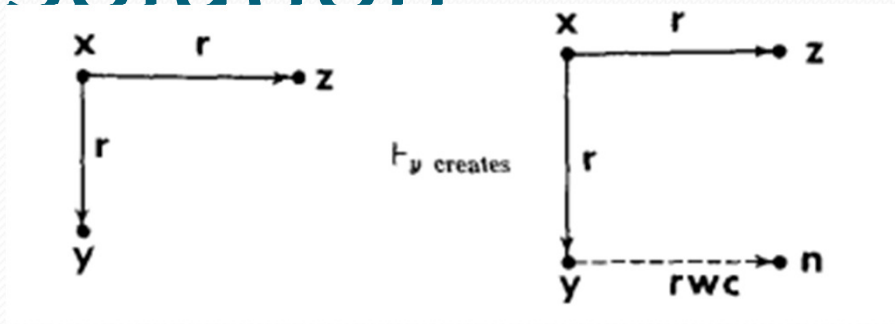


Example

- is it possible for y to read z?



Solution

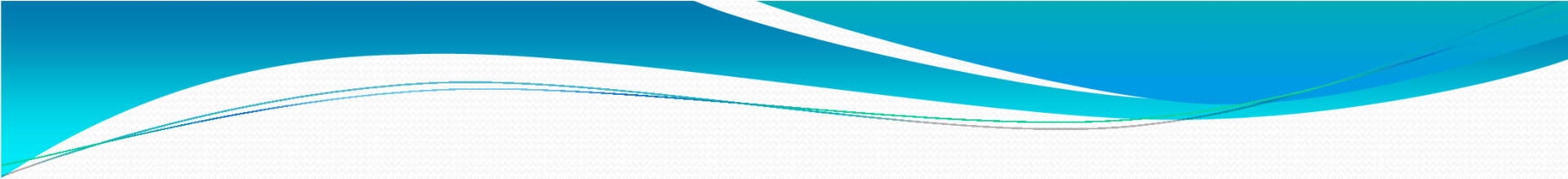


Summary



Conclusion

- So long as Denning's axioms are satisfied we will get a lattice-based information flow policy
- One-directional information flow in a lattice can be used for secrecy as well as for integrity but does not solve either problem completely
- To properly understand and enforce Information Security policies we must distinguish between
 - policy applied to users, and
 - policy applied to principals and subjects

- 
- Teşekkürler
 - Thank you
 - Efcharisto Poly
 - Muito Obrigado
 - Danke Schön
 - Bedankt
 - Labai Aciu