

LECTURE NOTES (02/01/2007)

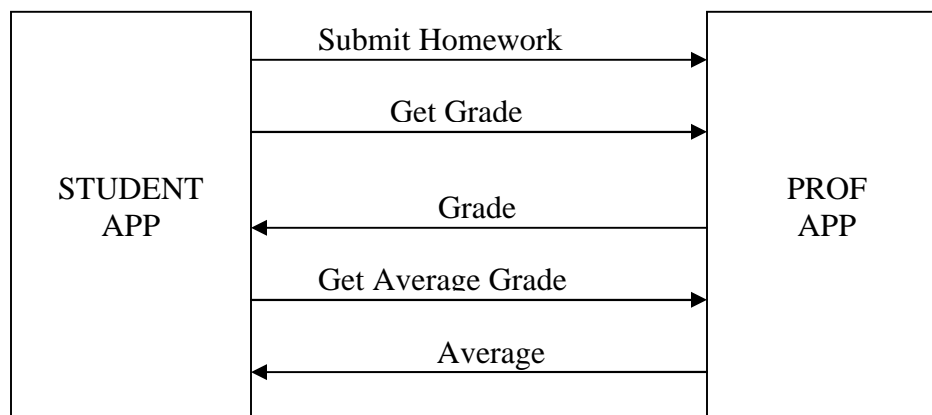
SUBMITTED BY: PRANAV MOOLWANEY (105954022)

ACCESS CONTROL

Mechanisms:

1. Access Control Matrices
 - ⇒ Capabilities
 - ⇒ Access Control Lists
2. Bell-Lapadula/BIBA/Lattice-Based Access Control
3. Rule Based Access Control (RBAC)

Consider the following example



- ⇒ Application can communicate via messages
- ⇒ OS tags each message with sender
- ⇒ Only student can access his/her own grade
- ⇒ Student can submit Homework only once

Thus as we can see, an Access Control System Consists of:

- ⇒ An Access Control Matrix
- ⇒ A Set of Rights
- ⇒ A Set of Commands

ACCESS CONTROL MATRIX

$$A_{d,o} = \{r \mid d \text{ has } r \text{ access to } o\}$$

	H/W queue	Stud.1 Grade	Stud.2 Grade	Class Avg
Professor	Dequeue Owner	Read Write Owner	Read Write Owner	Read Write Owner
Student 1	Enqueue	Read*		Read*
Student 2	Enqueue		Read*	Read*

RIGHTS

- ⇒ Owner [if owner belongs to $A_{d,o}$, then d can grade d' access on o]
- ⇒ Control [if control belongs to A_{d_1,d_2} , then d_1 can remove r access from $A_{d_2,o}$]
- ⇒ Copy [can delegate privileges]
- ⇒ Read
- ⇒ Write
- ⇒ Enqueue
- ⇒ Dequeue

Note: Positive V/S Negative Rules.

Example:

- ⇒ Let everyone from Accounting read file → Positive Rule
- ⇒ Do not let Bob Read a File → Negative Rule
- ⇒ But Bob is from Accounting → Which order should be given preference??

Rule of Thumb: Rules should be unordered.

Example: IP Tables in LINUX Kernel are ordered.

COMMANDS

It has the following format:

Command ($d_1, \dots, d_n, r_1, \dots, r_m, o_1, \dots, o_l$)

Example 1:

If (r_1' belongs to $A_{d_1',o_1'}$ and r_2' belongs to $A_{d_2',o_2'}$ and ...)

Op₁;

Op₂; /* Where these operators are adding or removing elements from an ACM */

.

.

.

Op_z;

Example 2:

Grant (d_1, d_2, r, o) /* d_1 is granting r access on o to d_2 */

If (owner $A_{d_1,o}$)

$A_{d_2,o} \cup = \{r\}$

Note: As per the theorem stated by Harrison, Puzzo and Ullman, it is undecidable to prove whether a sequence of commands will result in (r belongs to $A_{d,o}$)

IMPLEMENTING ACCESS CONTROL MATRICES

❖ **Capability Lists**

- Store ACM by row
- With each domain store a list of its access rights
- Easy to look up a domain's capabilities
- Hard to find all domains who can access an object
- Example: File handles

❖ Access Control Lists

- Store ACM by column
 - List everyone that can access an object with that object
 - Easy to see who can access something
 - Example: File Permissions, Windows NTFS, POSIX LINUX
-

BELL LAPADULA/BIBA/MULTILEVEL SECURITY

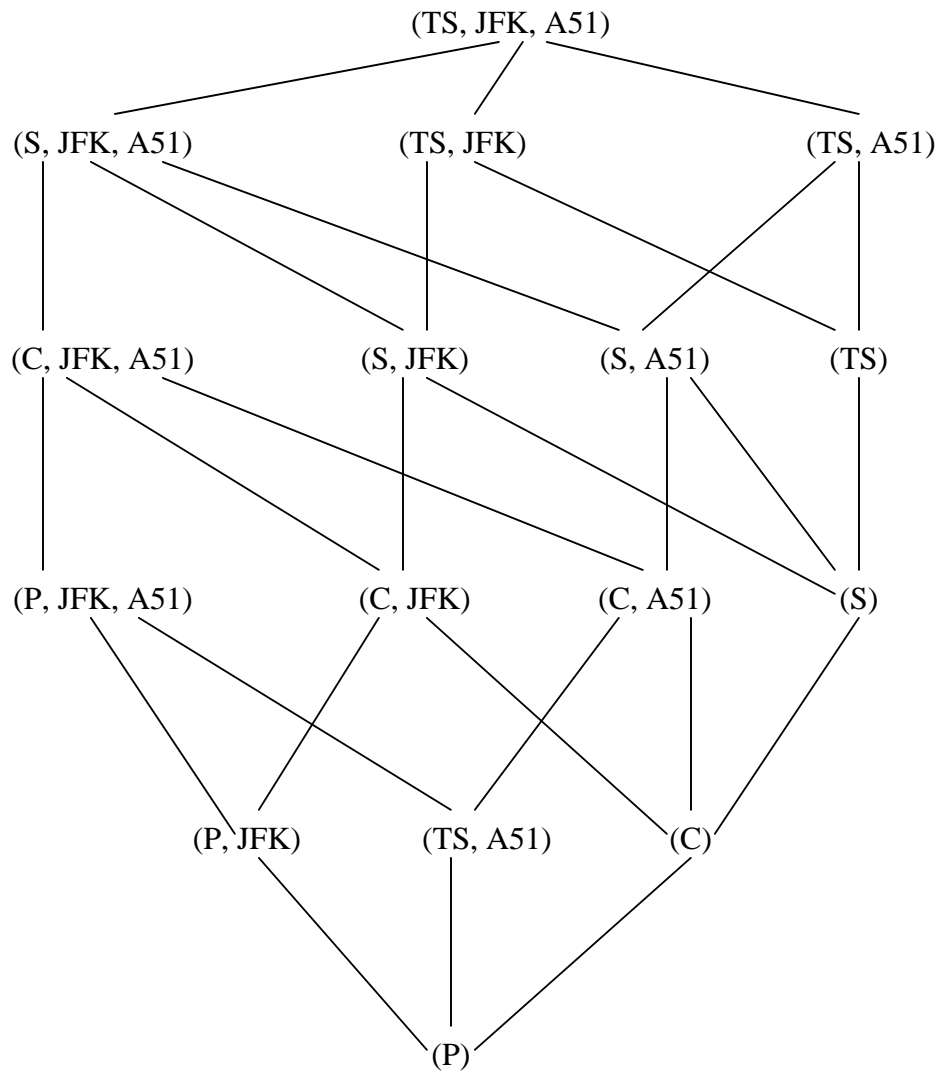
- ⇒ Inspired by Military Access Control
- ⇒ It deals with secrecy only [Bell]
- ⇒ BIBA deals with Integrity

Granularity: Top Secret → Secret → Confidential → Public

Examples of Compartments: Who Shot JFK? Area 51.

RULES

- If o is labeled with $C = \{C_1, C_2, \dots, C_n\}$ then domain must have access to all compartments in C .
- If o is labeled with secrecy level s , then domain d can only access o if d is labeled with $s' \geq s$.
- A label is a pair $l = (s, C)$ and, $(s_1, C_1) \leq (s_2, C_2)$ iff $(s_1 \leq s_2 \text{ and } C_2 \text{ contains } C_1)$



❖ Least Upper Bound

$LUB (l_1, l_2) = l$ such that $l_1 \leq l$ AND $l_2 \leq l$ AND for all l' such that $\{ l_1 \leq l' \text{ AND } l_2 \leq l' \text{ AND } l < l' \}$

Example:

(TS, JFK, A51) is bigger than (S, JFK, A51) AND (TS, JFK)

(S, JFK) is bigger than (P, JFK) AND (S)

❖ For Writes

Since we are only worried about secrecy and not integrity,

If d has a label l and o has a label l' then

d can write o if $l \leq l'$

AND d can read o if $l' \leq l$