

# IS 2150 / TEL 2810

## Introduction to Security



James Joshi  
Associate Professor, SIS

Lecture 3.2  
September 11, 2012

Access Control Model  
Foundational Results



# Protection System

---

- State of a system
  - Current values of
    - memory locations, registers, secondary storage, etc.
    - other system components
- Protection state (P)
  - A system state that is considered secure
- A protection system
  - Captures the conditions for state transition
  - Consists of two parts:
    - A set of generic rights
    - A set of commands



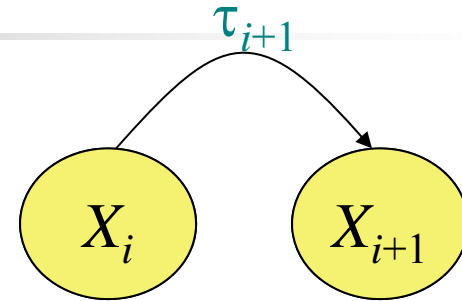
# Protection System

---

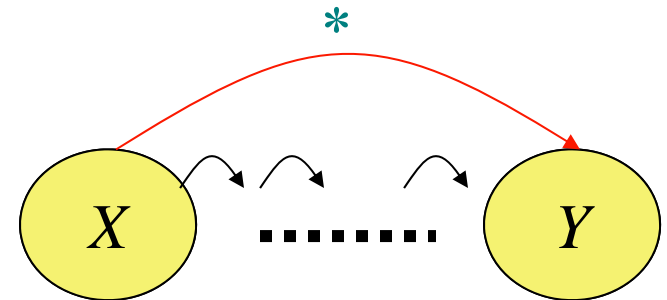
- Subject ( $S$ : set of all subjects)
  - Eg.: users, processes, agents, etc.
- Object ( $O$ : set of all objects)
  - Eg.: Processes, files, devices
- Right ( $R$ : set of all rights)
  - An action/operation that a subject is allowed/disallowed on objects
  - Access Matrix  $A$ :  $a[s, o] \subseteq R$
- Set of Protection States:  $(S, O, A)$ 
  - Initial state  $X_0 = (S_0, O_0, A_0)$

# State Transitions

$X_i \xrightarrow{\tau_{i+1}} X_{i+1}$  : upon transition  $\tau_{i+1}$ , the system moves from state  $X_i$  to  $X_{i+1}$

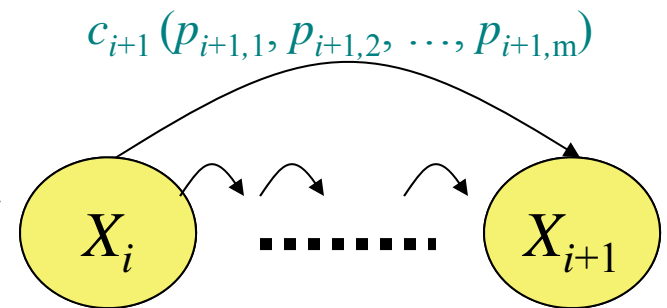


$X \xrightarrow{*} Y$  : the system moves from state  $X$  to  $Y$  after a set of transitions



$X_i \xrightarrow{c_{i+1}(p_{i+1,1}, p_{i+1,2}, \dots, p_{i+1,m})} X_{i+1}$  : state transition upon a command

For every command there is a sequence of state transition operations





# Primitive commands (HRU)

Create subject $s$	Creates new row, column in ACM; $s$ does not exist prior to this
Create object $o$	Creates new column in ACM $o$ does not exist prior to this
Enter $r$ into $a[s, o]$	Adds $r$ right for subject $s$ over object $o$ Ineffective if $r$ is already there
Delete $r$ from $a[s, o]$	Removes $r$ right from subject $s$ over object $o$
Destroy subject $s$	Deletes row, column from ACM;
Destroy object $o$	Deletes column from ACM



# Primitive commands (HRU)

---

Create subject  $s$

Creates new row, column in ACM;  
 $s$  does not exist prior to this

Precondition:  $s \notin S$

Postconditions:

$$S' = S \cup \{s\}, O' = O \cup \{s\}$$

$(\forall y \in O')[a'[s, y] = \emptyset]$  (row entries for  $s$ )

$(\forall x \in S')[a'[x, s] = \emptyset]$  (column entries for  $s$ )

$(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$



# Primitive commands (HRU)

Enter  $r$  into  $a[s, o]$

Adds  $r$  right for subject  $s$  over object  $o$

Ineffective if  $r$  is already there

Precondition:  $s \in S, o \in O$

Postconditions:

$$S' = S, O' = O$$

$$a'[s, o] = a[s, o] \cup \{ r \}$$

$$(\forall x \in S')(\forall y \in O')$$

$$[(x, y) \neq (s, o) \rightarrow a'[x, y] = a[x, y]]$$



# System commands

---

- [Unix] process  $p$  creates file  $f$  with owner  $own$  and  $read$  and  $write$  ( $r, w$ ) will be represented by the following:

Command  $create\_file(p, f)$

Create object  $f$

Enter  $own$  into  $a[p, f]$

Enter  $r$  into  $a[p, f]$

Enter  $w$  into  $a[p, f]$

End





# System commands

---

- Process  $p$  creates a new process  $q$

Command *spawn\_process*( $p, q$ )

Create subject  $q$ ;

Enter *own* into  $a[p, q]$

Enter  $r$  into  $a[p, q]$

Enter  $w$  into  $a[p, q]$

Enter  $r$  into  $a[q, p]$

Enter  $w$  into  $a[q, p]$

End

← Parent and child can  
signal each other



# System commands

---

- Defined commands can be used to update ACM

Command *make\_owner(p, f)*

Enter *own* into *a[p,f]*

End

- Mono-operational:
  - the command invokes only one primitive



# Conditional Commands

---

- Mono-operational + mono-conditional

Command *grant\_read\_file(p, f, q)*

If *own* in  $a[p, f]$

Then

Enter *r* into  $a[q, f]$

End



# Conditional Commands

---

- Mono-operational + biconditional

Command *grant\_read\_file(p, f, q)*

If  $r$  in  $a[p, f]$  and  $c$  in  $a[p, f]$

Then

Enter  $r$  into  $a[q, f]$

End

- Why not "OR"??



# Fundamental questions

---

- How can we determine that a system is secure?
  - Need to define what we mean by a system being “secure”
- Is there a generic algorithm that allows us to determine whether a computer system is secure?



# What is a secure system?

---

- A simple definition
  - A secure system doesn't allow violations of a security policy
- Alternative view: based on distribution of rights
  - **Leakage of rights:** (unsafe with respect to right  $r$ )
    - Assume that  $A$  representing a secure state does not contain a right  $r$  in an element of  $A$ .
    - A right  $r$  is said to be leaked, if a sequence of operations/commands adds  $r$  to an element of  $A$ , which did not contain  $r$



# What is a secure system?

---

- Safety of a system with initial protection state  $X_0$ 
  - Safe with respect to  $r$ : System is *safe with respect to  $r$*  if  $r$  can never be leaked
  - Else it is called *unsafe with respect to right  $r$* .