

Chapter 2: Access Control Matrix

- Overview
- Access Control Matrix Model
- Protection State Transitions
 - Commands
 - Conditional Commands

Overview

- Protection state of system
 - Describes current settings, values of system relevant to protection
- Access control matrix
 - Describes protection state precisely
 - Matrix describing rights of subjects
 - State transitions change elements of matrix

Description

objects (entities)

	O_1	...	O_m	S_1	...	S_n
S_1						
S_2						
...						
S_n						

subjects

- Subjects $S = \{ s_1, \dots, s_n \}$
- Objects $O = \{ o_1, \dots, o_m \}$
- Rights $R = \{ r_1, \dots, r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \dots, r_y \}$
means subject s_i has rights r_x, \dots, r_y over object o_j

Example 1

- Processes p, q
- Files f, g
- Rights r, w, x, a, o

	f	g	p	q
p	rwo	r	$rwxo$	w
q	a	ro	r	$rwxo$

Example 2

- Procedures *inc_ctr*, *dec_ctr*, *manage*
- Variable *counter*
- Rights *+*, *-*, *call*

	<i>counter</i>	<i>inc_ctr</i>	<i>dec_ctr</i>	<i>manage</i>
<i>inc_ctr</i>	<i>+</i>			
<i>dec_ctr</i>	<i>-</i>			
<i>manage</i>		<i>call</i>	<i>call</i>	<i>call</i>

State Transitions

- Change the protection state of system
- \vdash represents transition
 - $X_i \vdash_{\tau} X_{i+1}$: command τ moves system from state X_i to X_{i+1}
 - $X_i \vdash^* X_{i+1}$: a sequence of commands moves system from state X_i to X_{i+1}
- Commands often called *transformation procedures*

Primitive Operations

- **create subject s ; create object o**
 - Creates new row, column in ACM; creates new column in ACM
- **destroy subject s ; destroy object o**
 - Deletes row, column from ACM; deletes column from ACM
- **enter r into $A[s, o]$**
 - Adds r rights for subject s over object o
- **delete r from $A[s, o]$**
 - Removes r rights from subject s over object o

Creating File

- Process p creates file f with r and w permission

```
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
```


Mono-Operational Commands

- Make process p the owner of file g

```
command make • owner( $p$ ,  $g$ )  
    enter own into  $A[p, g]$ ;  
end
```

- Mono-operational command
 - Single primitive operation in this command

Conditional Commands

- Let p give q r rights over f , if p owns f
command $grant \cdot read \cdot file \cdot 1(p, f, q)$
 if own **in** $A[p, f]$
 then
 enter r **into** $A[q, f];$
 end
- Mono-conditional command
 - Single condition in this command

Multiple Conditions

- Let p give q r and w rights over f , if p owns f and p has c rights over q

```
command grant.read.file.2( $p, f, q$ )  
    if own in  $A[p, f]$  and  $c$  in  $A[p, q]$   
    then  
        enter  $r$  into  $A[q, f]$ ;  
        enter  $w$  into  $A[q, f]$ ;  
end
```

Key Points

- Access control matrix simplest abstraction mechanism for representing protection state
- Transitions alter protection state
- 6 primitive operations alter matrix
 - Transitions can be expressed as commands composed of these operations and, possibly, conditions