

**CS590U**  
**Access Control: Theory and Practice**

Lecture 3 (September 2<sup>nd</sup>)  
Access Control Matrix Models and  
The Take-Grant Model

**The HRU Access Matrix Model**



## Protection Systems

---

- A protection system has
  - a finite set  $R$  of generic rights
  - a finite set of commands

3



## The State of A Protection System

---

- A set  $O$  of objects
- A set  $S$  of subjects that is a subset of  $O$
- An access control matrix
  - one row for each subject
  - one column for each object
  - each cell contains a set of rights

4



## Commands

- A command has the form

command  $a(X_1, X_2, \dots, X_k)$

if

$r_1$  in  $(X_{s1}, X_{o1})$  and ... and  $r_m$  in  $(X_{sm}, X_{om})$

then

$op_1 \dots op_n$

end

5



## Primitive Operations

- enter  $r$  into  $(X_s, X_o)$ 
  - Condition:  $X_s \in S$  and  $X_o \in O$
  - $r$  may already exist in  $(X_s, X_o)$
- delete  $r$  from  $(X_s, X_o)$ 
  - Condition:  $X_s \in S$  and  $X_o \in O$
  - $r$  does not need to exist in  $(X_s, X_o)$

6



## Primitive Operations

- create subject  $X_s$ 
  - Condition:  $X_s \notin O$
- create object  $X_o$ 
  - Condition:  $X_o \notin O$
- delete subject  $X_s$ 
  - Condition:  $X_s \in S$
- delete object  $X_o$ 
  - Condition:  $X_o \in O$  and  $X_o \notin S$

7



## Example 4 in [HRU]:

- **Problem:** how to Implementing Unix access control in HRU
- **Difficulty:** the owner of a file may specify the privileges of all other users
- **Solution:** the cell  $(f,f)$  determines who can access the file  $f$
- **Question:** anything to say about this solution? other solutions?

8



## The Safety Problem

---

- What do we mean by “safe”?
  - Definition 1: “access to resources without the **concurrency** of the owner is impossible”
  - Definition 2: “the user should be able to tell whether what he is about to do (give away a right, presumably) can lead to the further leakage of that right to **truly unauthorized** subjects”

9



## Defining the Safety Problem

---

- “Suppose a subject  $s$  plans to give subjects  $s'$  generic right  $r$  to object  $o$ . The natural question is whether the current access matrix, with  $r$  entered into  $(s',o)$ , is such that generic right  $r$  could subsequently be entered somewhere new.”

10



## Defining the Safety Problem

- To avoid a trivial “unsafe” answer because  $s$  himself can confer generic right  $r$ , we should in most circumstances **delete**  $s$  itself from the matrix. It might also make sense to **delete** from the matrix any other “reliable” subjects who could grant  $r$ , but whom  $s$  “trusts” will not do so.

11



## Defining the Safety Problem

- It is only by using the hypothetical safety test in this manner, with “reliable” subjects deleted, that the ability to test whether a right can be leaked has a useful meaning in terms of whether it is safe to grant a right to a subject.

12



## The Safety Problem is not Well-Motivated!

- Removing trusted subjects is a problem.
  - **why:** also remove possible attacks
  - **source of the problem:** no concept of initiator of a command. Without it, cannot define **concurrency** or **truly untrusted**.
- Identify a single subject as untrusted is not enough.
  - more sophisticated safety queries are needed

13



## Let Us Look at the Mathematical Problem Anyway

- Given a protection system, a state of the system, determines whether a right could appear in a cell
- Undecidable in the general case

14



## Simulating Turing Machines using Protection Systems

- The set of generic rights include
  - the states and tape symbols of the Turing machine,
  - and two special rights: `own', `end'
- Turing Machine instructions are mapped to commands

15



## Mapping a Tape to an Access Matrix

- The  $j$ 'th cell on the tape = the subject  $s_j$
- The  $j$ 'th cell has symbol  $X \Rightarrow X \in (s_j, s_j)$
- The head is at the  $j$ 'th cell and the current state is  $q \Rightarrow q \in (s_j, s_j)$
- The  $k$ 'th cell is the last  $\Rightarrow$   
`end'  $\in (s_k, s_k)$
- For  $1 \leq j < k$ , `own'  $\in (s_j, s_{j+1})$

16

## Moving Left:

$(q, X) \rightarrow (p, Y, \text{left})$

```
command  $C_{qX}(s, s')$ 
  if  $q$  in  $(s', s')$  and  $X$  in  $(s', s')$ 
    and `own' in  $(s, s')$ 
  then delete  $q$  from  $(s', s')$ 
    delete  $X$  from  $(s', s')$ 
    enter  $Y$  into  $(s', s')$ 
    enter  $p$  into  $(s, s)$ 
end
```


17

## Moving Right (case one):

$(q, X) \rightarrow (p, Y, \text{right})$

```
command  $C_{qX}(s, s')$ 
  if  $q$  in  $(s, s)$  and  $X$  in  $(s, s)$ 
    and `own' in  $(s, s')$ 
  then delete  $q$  from  $(s, s)$ 
    delete  $X$  from  $(s, s)$ 
    enter  $Y$  into  $(s, s)$ 
    enter  $p$  into  $(s', s')$ 
end
```

18



## Moving Right (case two): (q, X) -> (p, Y, right)

```
command  $C_{qX}(s, s')$ 
  if q in (s, s) and X in (s, s)
    and `end` in (s, s)
  then delete q from (s, s)   delete X from (s, s)
     enter Y into (s, s)
     create subject  $s'$        enter `own` into ( $s'$ ,  $s'$ )
     enter p into ( $s'$ ,  $s'$ )   enter B into ( $s'$ ,  $s'$ )
     delete end from (s, s)   enter `end` into ( $s'$ ,  $s'$ )
  end
```

19



## Other Results

- The safety question is
  - decidable for mono-operational
  - PSPACE-complete for systems without create
  - undecidable for biconditional monotonic protection systems
  - decidable for monoconditional monotonic protection systems

20



## The Take-Grant Model

---

- Two special rights `take' and `grant'
- The state is represented by a graph
- **The take rule:** if  $x$  has `take' right over  $z$ , and  $z$  has right  $r$  over  $y$ , then  $x$  can get right  $r$  over  $y$
- **The grant rule:** if  $z$  has `grant' right over  $x$ , and  $z$  has right  $r$  over  $y$ , then  $x$  can get right  $r$  over  $y$

21



## The Take and the Grant Rule

---

- **The take rule:** if  $x$  has `take' right over  $z$ , and  $z$  has right  $r$  over  $y$ , then  $x$  can get right  $r$  over  $y$
- **The grant rule:** if  $z$  has `grant' right over  $x$ , and  $z$  has right  $r$  over  $y$ , then  $x$  can get right  $r$  over  $y$

22



## Other Models

---

- Schematic Protection Model
- Typed Access Matrix Model
  - developed by Ravi Sandhu, et al.

23



## Discretionary Access Control

---

- Discretionary Access Control
  - a means of restricting access to objects based on the identity of subjects and/or groups to which they belong
  - a subject with a certain access authorization may, at its discretion, be capable of passing (perhaps indirectly) that access type or a subset on to any other subject

24



## Problems with DAC

---

- No semantic information associated with objects
- Arbitrary propagation of rights
- No control on information flow
  - one can make a copy of a piece of data and make it available to others
- Trojan horse problem

25



## Non-discretionary Access Control

---

- Right propagation is not discretionary
- Many possibilities

26



## Mandatory Access Control

---

- Each subject is associated with a clearance
- Each object is associated with a sensitivity level
- Access authorization is based on the clearance and the sensitivity level
- Centrally administered right management

27



## Role-Based Access Control

---

- A form of non-discretionary access control
- Roles add a level of indirection between subjects and objects
- Centrally administered right management

28



## End of Lecture 3

---

- Pre-proposal due on the beginning of next class