Access Control Matrix and Safety Results

CS461/ECE422 Computer Security I, Fall 2009

Based on slides provided by Matt Bishop for use with

Computer Security: Art and Science

Plus HRU examples from Ravi Sandhu

Reading

- Chapter 2 Access Control Matrix
- A little bit from Chapter 3 to talk about Safety

Outline

- Motivation
- Access Control Matrix Model
- Protection State Transitions
- HRU Model
 - Commands
 - Conditional Commands
- Basic Safety results

Motivation

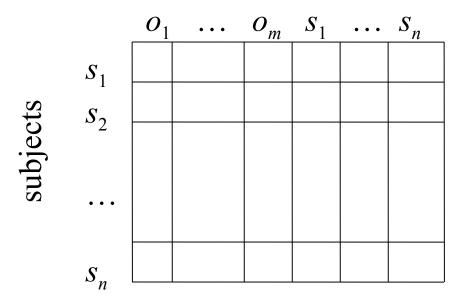
- Access Control Matrix (ACM) and related concepts provides very basic abstraction
 - Map different systems to a common form for comparison
 - Enables standard proof techniques
 - Not directly used in implementation
- Basis for key safety decidability results

Definitions

- 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)



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

	counter	inc_ctr	dec_ctr	manage
inc_ctr	+			
dec_ctr	_			
manage		call	call	call

Boolean Expression Evaluation

- ACM controls access to database fields
 - Subjects have attributes
 - Verbs define type of access
 - Rules associated with objects, verb pair
- Subject attempts to access object
 - Rule for object, verb evaluated, grants or denies access

Example

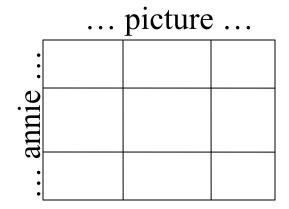
- Subject annie
 - Attributes role (artist), groups (creative)
- Verb paint
 - Default 0 (deny unless explicitly granted)
- Object picture
 - Rule:
 paint: 'artist' in subject.role and 'creative' in subject.groups and time.hour > 0 and time.hour < 5

ACM at 3AM and 10AM

At 3AM, time condition met; ACM is:

paint

At 10AM, time condition not met; ACM is:



History

Query-Set overlap limit = 2

Database:

name	position	age	salary
Alice	teacher	45	\$40,000
Bob	aide	20	\$20,000
Carol	principal	37	\$60,000
Dave	teacher	50	\$50,000
Eve	teacher	33	\$50,000

Queries:

C1: sum(salary, "position = teacher") = 140,000

C2: count(set(age < 40 & position = teacher)

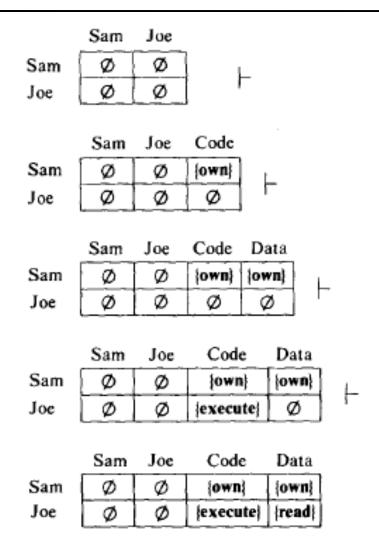
C3: sum(salary, "age > 40 & position = teacher") should not be answered (deduce Eve's salary)

Slide #2-12

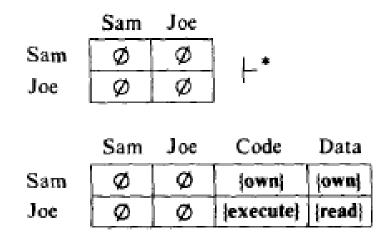
State Transitions

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

Example Transitions



Example Composite Transition



HRU Model

- Harrison, Ruzzo, and Ullman proved key safety results in 1976
- Talked about systems
 - With initial protection state expressed in ACM
 - State transition commands built from a set of primitive operations
 - Applied conditionally.

HRU Commands and Operations

• command $\alpha(X1, X2, \ldots, Xk)$

if rl in A[Xs1, Xo1] and r2 in A[Xs2, Xo2] and ... rk in A[Xsk, Xok]

then

op1; op2; ... opn

end

- 6 Primitive Operations
 - enter r into A[Xs, Xo]
 - delete r from A[Xs, Xo]
 - create subject Xs
 - create object Xo
 - destroy subject Xs
 - destroy object Xo

Create Subject

- Precondition: $s \notin S$
- Primitive command: create subject s
- Postconditions:

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

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

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

Create Object

- Precondition: $o \notin O$
- Primitive command: create object o
- Postconditions:

$$-S' = S, O' = O \cup \{o\}$$

$$-(\forall x \in S')[a'[x, o] = \emptyset]$$

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

Add Right

- Precondition: $s \in S$, $o \in O$
- Primitive command: enter r into a[s, o]
- Postconditions:

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

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

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

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

Delete Right

- Precondition: $s \in S$, $o \in O$
- Primitive command: **delete** r **from** a[s, o]
- Postconditions:

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

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

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

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

Destroy Subject

- Precondition: $s \in S$
- Primitive command: destroy subject s
- Postconditions:

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

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

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

Destroy Object

- Precondition: $o \in O$
- Primitive command: destroy object o
- Postconditions:

$$-S' = S, O' = O - \{o\}$$

$$-(\forall x \in S')[a'[x, o] = \emptyset]$$

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

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
```

Confer Right

- Example of a mono-conditional command
- Also, mono-operational command

```
command confer_r(owner, friend,f)
  if own in A[owner, f]
  then enter r into A[friend,f]
end
```

Remove Right

Example using multiple conditions

```
• command remove_r(owner,exfriend, f)
  if own in A[owner, f] and
    r in A[exfriend, f]
    then delete r from A[exfriend, f]
  end
```

Copy Right

- Allows possessor to give rights to another
- Often attached to a right, so only applies to that right
 - -r is read right that cannot be copied
 - -rc is read right that can be copied
- Is copy flag copied when giving r rights?
 - Depends on model, instantiation of model

Attenuation of Privilege

- Principle says you can't give rights you do not possess
 - Restricts addition of rights within a system
 - Usually *ignored* for owner
 - Why? Owner gives herself rights, gives them to others, deletes her rights.

The Safety Problem

- Given
 - initial state
 - protection scheme (HRU commands)
- Can r appear in a cell that exists in the initial state and does not contain r in the initial state?
- More specific question might be: can r appear in a specific cell A[s,o]

Safety with respect to r

Safety of a Specific Access Control System

- Is it decidable?
- Is it computationally feasible?

- Safety is undecidable in the general HRU model
 - Maps to the Halting problem

Safety Results

- Constraints on HRU help some
 - Safety for mono-operational systems is decidable but NP-Complete
 - Mono-conditional monotonic HRU is decidable but not interesting
- Other systems proposed with better results
 - Take-Grant model decidable in linear time
- Still an active research area
 - Comparing expressiveness with safety

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
- Early safety proofs build on this HRU model