Access Control Matrix

- Overview
- Access Control Matrix Model
 - Boolean Expression Evaluation
 - History
- Protection State Transitions
 - Commands
 - Conditional Commands
- Special Rights
 - Principle of Attenuation of Privilege

Overview

• State of a system

It is the collection of the current values of all memory locations, all secondary storage, all registers and other components of the system.

- Protection state of a system
 - The subset of this collection that deals with protection.
 - Describes current settings, values of system relevant to protection.

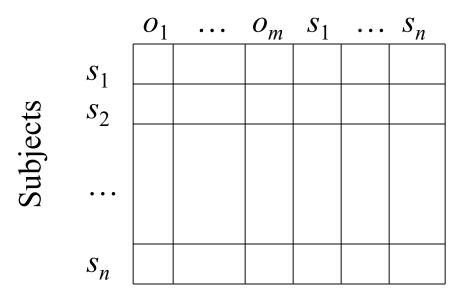
Access Control Matrix (ACM)

• ACM

- Describes protection state precisely.
- Matrix describing rights of subjects.
- State transitions change elements of matrix.
- P The set of possible protection states.
- $Q \subseteq P$ Those states in which the system is authorized to reside.
- (P-Q) System is not secure.

Description

Objects (entities)



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

	f	\boldsymbol{g}	p	q
p	rwe	r	rwe	\mathcal{W}
q	а	re	r	rwe

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:

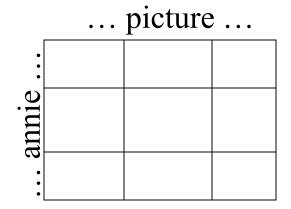
Paint: 'artist' in subject.role and 'creative' in subject.groups and time.hour ≤ 0 and time.hour < 5

ACM at 3AM and 10AM

At 3AM, time condition met; ACM is:

paint

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



Access Control By History

- Assume that the database contains N records.
- Users query the database about sets of records *C*; this set is the *query set*.
- The goal of attackers is to obtain a statistic for an individual record.
- The *query-set-overlap control* is a prevention mechanism that answers queries only when the size of the intersection of the query set and each previous query set is smaller than some parameter.

History

Database: Set=2

Name	Position	Age	Salary
Alice	Teacher	45	\$40,000
Bob	Aide	20	\$20,000
Cathy	Principal	37	\$60,000
Dilbert	Teacher	50	\$50,000
Eve	Teacher	33	\$50,000

Queries:

- 1.sum(salary, "Position = Teacher") = 140,000
- 2.sum(salary, "Age < 40 & Position = Teacher")
- 3. sum(salary, "Age > 40 & Position = Teacher") should not be answered (deduce Eve's salary)

ACM of Database Queries

```
O_i = \{ \text{Union of the objects referenced in query } i \}
F(O_i) = \{ \text{read} \} \qquad |O_i| > 1
F(O_i) = \emptyset
                          otherwise
Element of the matrix A[s, o] = F(O_i \cap \{o\}) for query i,
where 1 \le i and O_0 = \emptyset
C1: A[asker, (Alice, Dilbert, Eve)] = { read } and |O_1| = 3
C2: A[asker, Eve] = {read} and |O_2| = 1
C3: A[asker, (Alice, Dilbert)] = \emptyset and |O_3| = 2
```

ACM of Database Queries

```
O_i = \{\text{objects referenced in query } i\}
F(o_i) = \{\text{read}\} for o_i in O_i, if |\bigcup_{j=1,...,i} O_j| > 1
F(o_i) = \emptyset
                          for o_i \in O_i, otherwise
1. O_1 = \{\text{Alice, Dilbert, Eve}\}\ and no previous query set,
      so:
           A[asker, Alice] = f(Alice) = \{read\}
           A[asker, Dilbert] = f(Dilbert) = \{read\}
           A[asker, Eve] = f(Eve) = {read}
      and query can be answered.
```

But Query 2

From last slide:

$$F(o_i) = \{ \text{ read } \}$$
 for o_j in O_i , if $|\bigcup_{j=1,...,i} O_j| > 1$
 $F(o_i) = \emptyset$ for o_j in O_i , otherwise

2. $O_2 = \{ \text{ Alice, Dilbert } \} \text{ but } | O_2 \cap O_1 | = 2 \text{ so}$ $A[\text{asker, Alice}] = f(\text{Alice}) = \emptyset$ $A[\text{asker, Dilbert}] = f(\text{Dilbert}) = \emptyset$ and query cannot be answered.

State Transitions

- Change the protection state of system
- | represents transition
 - $-X_i \mid -_{\tau} X_{i+1}$: 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

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

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 owner 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 • read • file • 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
```

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

Own Right

- Usually allows possessor to change entries in ACM column
 - So owner of object can add, delete rights for others
 - May depend on what system allows
 - Can't give rights to specific (set of) users
 - Can't pass copy flag to specific (set of) users

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 himself/herself rights, gives them to others, deletes his/her rights.

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.