Access Control Matrix

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

Access Control and Authorization

- *Access control is a process to determine "Who does what to what," based on a policy.*
- It is controlling access of who gets in and out of the system and who uses what resources, when, and in what amounts.
- Access control is restricting access to a system or system resources based on something other than the identity of the user

Access Operations

- Unix

	file	directory
read	read from a file	list directory contents
write	write to a file	create or rename a file in a directory
execute	execute a (program) file	search the directory

Access rights specific to a file are changed by my modifying the file's entry in its directory

Access Operations

- Windows NT

  Permissions of Windows New Technology File System (NTFS)
  - read
  - write
  - execute
  - delete
  - change permission
  - change ownership

Protection State

Definition (Protection State)

The state of a system is the collection of the current values of all memory locations, all secondary storage, and all registers and other components of the system. The subset of this collection that deals with protection is the protection state of the system.

Definition (Access Control Matrix)

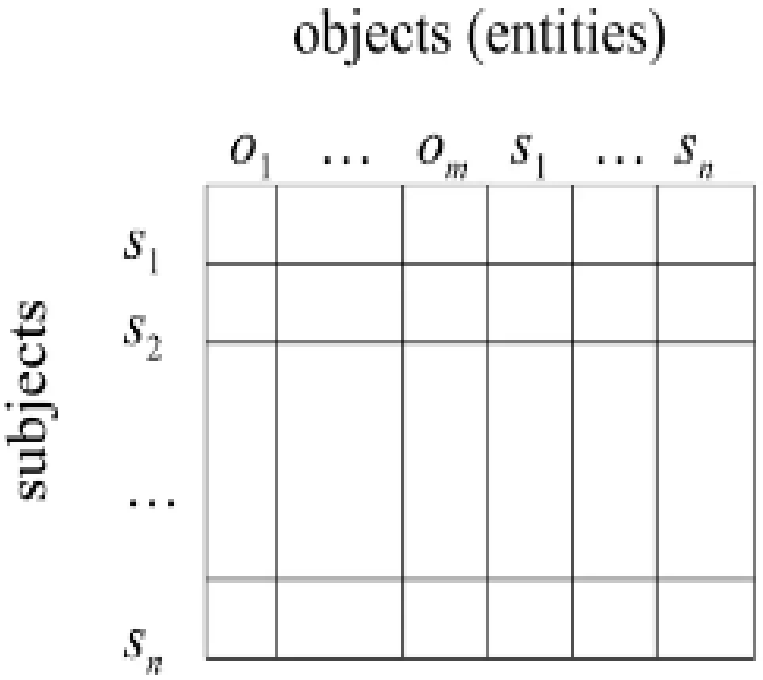
An access control matrix is a tool that can describe the current protection state.

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*

Description



- Subjects $S = \{ s_1,\ldots,s_x \}$
- Objects $O = \{ o_1,\ldots,o_z \}$
- Rights $R = \{ r_1,\ldots,r_t \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ means subject s_i has rights $r_x, \ldots, r_y$ over object o_j

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

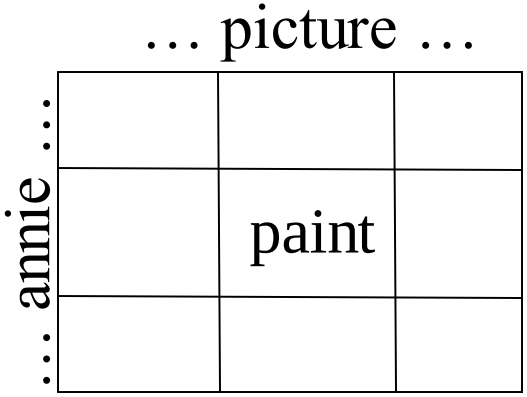
Example

- Subject annie
  - Attributes role (artist), groups (creative)
- Verb paint
  - Default 0 (deny unless explicitly granted)
- Object picture
  - Rule:

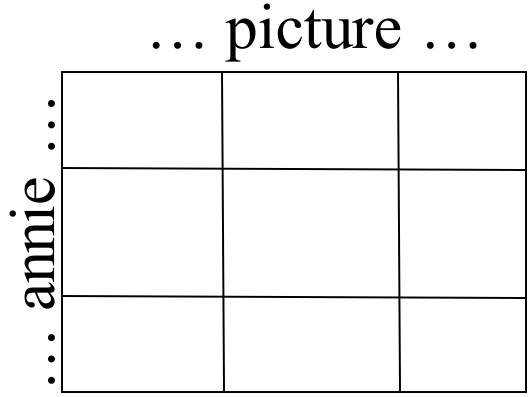
paint: ‘artist’ in subject.role and
‘creative’ in subject.groups and
time.hour ≥ 0 and time.hour < 5

ACM at 3AM and 10AM

At 3AM, time condition met; ACM is:



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



Access Controlled by History

- Statistical databases are designed to answer queries about groups of records yet not reveal information about any single specific record
- Assume that a 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 r

Access Controlled by History

Exercise

Consider the following database and set $r = 2$

name	position	age	salary
Celia	teacher	45	$40,000
Heidi	aide	20	$20,000
Holly	principal	37	$60,000
Leonard	teacher	50	$50,000
Matt	teacher	33	$50,000

ACM of Database Queries

O_i = { objects referenced in query i }

$f(o_j)$ = { read } for $o_j \in O_i$, if $|\cup_{j=1,\ldots,i} O_j| < 2$

$f(o_j) = \varnothing$ for $o_j \in O_i$, otherwise

1. O_1 = { Celia, Leonard, Matt} and no previous query set, so:

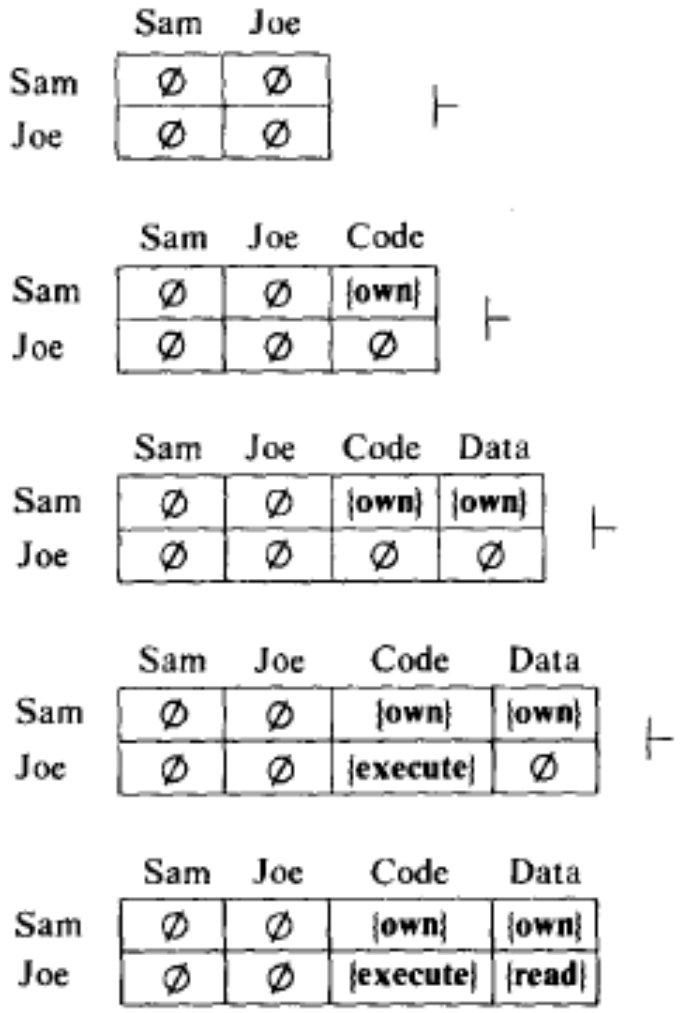
   A[asker, Celia, Leonard, Matt] = { read }

   and query can be answered

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*

Example Transitions



Example Composite Transition

	Sam	Joe
Sam	∅	∅
Joe	∅	∅

$\vdash^*$

	Sam	Joe	Code	Data
Sam	∅	∅	{own}	{own}
Joe	∅	∅	{execute}	{read}

Create Subject

- Precondition: $s \notin S$
- Primitive command: **create subject** s
- Postconditions:
  - $S' = S \cup \{ s \}$, $O' = O \cup \{ s \}$
  - $(\forall y \in O')[a'[s, y] = \varnothing]$, $(\forall x \in S')[a'[x, s] = \varnothing]$
  - $(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$

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

Destroy Object

- Precondition: $o \in O$
- Primitive command: **destroy object** *o*
- Postconditions:
  - $S' = S, O' = O - \{ o \}$
  - $(\forall x \in S')[a'[x, o] = \varnothing]$
  - $(\forall x \in S')(\forall y \in O')\ [a'[x, y] = a[x, y]]$

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