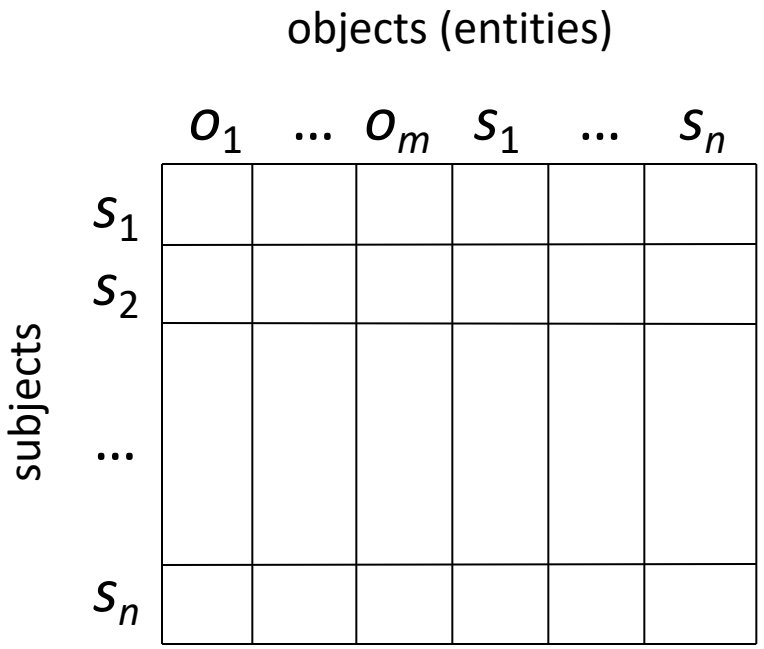
ECS 235B Module 4
Access Control Matrix

Description



- Subjects $S = \{ s_1, \ldots, s_n \}$
- Objects $O = \{ o_1, \ldots, o_m \}$
- Rights $R = \{ r_1, \ldots, r_k \}$
- 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

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

- Host names *telegraph*, *nob*, *toadflax*
- Rights *own*, *ftp*, *nfs*, *mail*

	telegraph	*nob*	*toadflax*
telegraph	*own*	*ftp*	*ftp*
nob		*ftp, mail, nfs, own*	*ftp, nfs, mail*
toadflax		*ftp, mail*	*ftp, mail, nfs, own*

Example 3

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

	counter	*inc_ctr*	*dec_ctr*	*manage*
inc_ctr	*+*			
dec_ctr	*–*			
manager		*call*	*call*	*call*

UNIX/Linux Access Controls

- Files: A is ~bishop/a.out (0755), B is /etc/passwd (0644), H is /home/bishop (0711), S is /bin/su (4711)

	A	*B*	*S*	*H*
bishop	*rwxo*	*r*	*x*	*rwxo*
zheng	*rx*	*r*	*x*	*x*
root	*rwx*	*rwo*	*rwxo*	*rwx*

UNIX/Linux Access Controls

- Access control matrices are dynamic:
- After bishop executes `chmod 700 /home/bishop`:

	A	*B*	*S*	*H*
bishop	*rwxo*	*r*	*x*	*rwxo*
muwei		*r x*		
root	*rwx*	*rwo*	*rwxo*	*rwx*

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

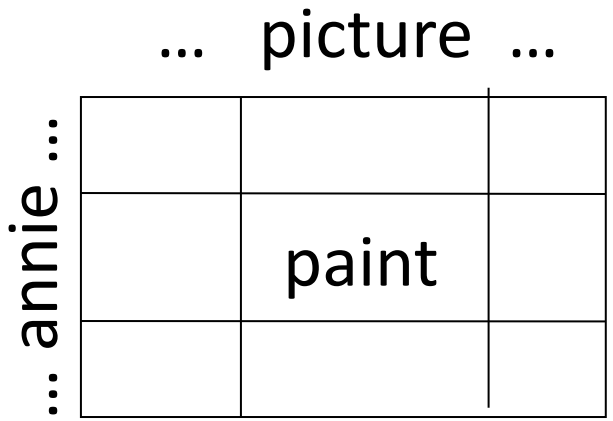
Example

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

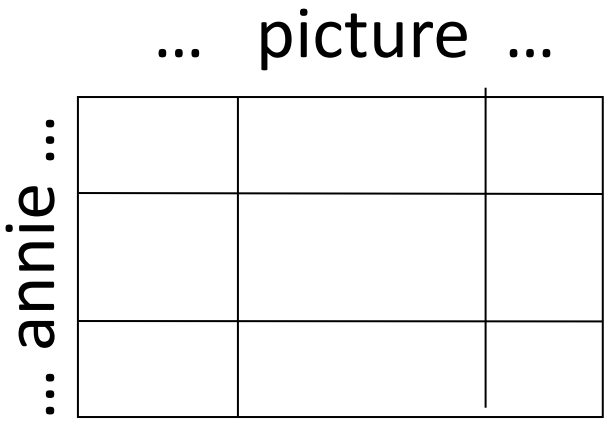
    paint: ‘artist’ in subject.role and
    ‘creative’ in subject.groups and
    time.hour ≥ 0 and time.hour ≤ 4

ACM at 3AM and 10AM

At 3AM, time condition met
ACM is:



At 10AM, time condition not met
ACM is:



History

- Problem: what a process has accessed may affect what it can access now
- Example: procedure in a web applet can access other procedures depending on what procedures it has already accessed
  - *S* set of *static rights* associated with procedure
  - *C* set of current rights associated with each executing process
  - When process calls procedure, rights are $S \cap C$

Example Program

```
// This routine has no filesystem access rights
// beyond those in a limited, temporary area
procedure helper_proc()
        return sys_kernel_file

// But this has the right to delete files
program main()
        sys_load_file(helper_proc)
        tmp_file = helper_proc()
        sys_delete_file(tmp_file)
```

- *sys_kernel_file* contains system kernel
- *tmp_file* is in limited area that *helper_proc*() can access

Before *helper_proc* Called

- Static rights of program

	sys_kernel_file	*tmp_file*
main	delete	delete
helper_proc		delete

- When program starts, current rights:

	sys_kernel_file	*tmp_file*
main	delete	delete
helper_proc		delete
process	delete	delete

After *helper_proc* Called

- Process rights are intersection of static, previous "current" rights:

	sys_kernel_file	*tmp_file*
main	delete	delete
helper_proc		delete
process		delete

Quiz

In an access control matrix, do the rights “r”, “w”, and “x” represent “read”, “write”, and “execute” permissions, respectively?

- Yes, because the permission symbols are tied to those permissions (“r” for “read”, “w” for “write”, “x” for “execute”).
- Possibly; the meanings of the permission symbols depends upon the instantiation.
- No, because the meanings of the permission symbols changes as the matrix evolves, so you cannot say what the symbols mean; you can only manipulate them based on the given commands.