Lecture 17: Discretionary Access Control

CS 181S

Fall 2020

Where we were…

- **Authentication:** mechanisms that bind principals to actions
- **Authorization:** mechanisms that govern whether actions are permitted
- **Audit:** mechanisms that record and review actions

Access Control Policy

- An **access control policy** specifies which of the **operations** associated with any given **object** each **principal** is authorized to perform
- Expressed as a relation ***Auth***:

Auth		Objects	
		dac.tex	dac.pptx
principals	ebirrell	r,w	r,w
	drdave	r	r
	studenta		r

Who defines authorizations?

- **Discretionary Access Control:** owner defines authorizations
- **Mandatory Access Control:** centralized authority defines authorizations

Access Control Mechanisms

- A **reference monitor** is consulted whenever one of a predefined set operations is invoked
  - operation $\langle P, O, op \rangle$ is allowed to proceed only if the invoker P is authorized to perform op on object O
- Can enforce **confidentiality** and/or **integrity**
- **Assumption:** Predefined operations are the sole means by which principals can learn or update information.
- **Assumption:** All predefined operations can be monitored (complete mediation).

Design Principles

- **Principle of Failsafe Defaults** favors defining an access control policy by enumerating privileges rather than prohibitions.
- **Principle of Least Privilege** is best served by having fine-grained principals, objects, and operations.

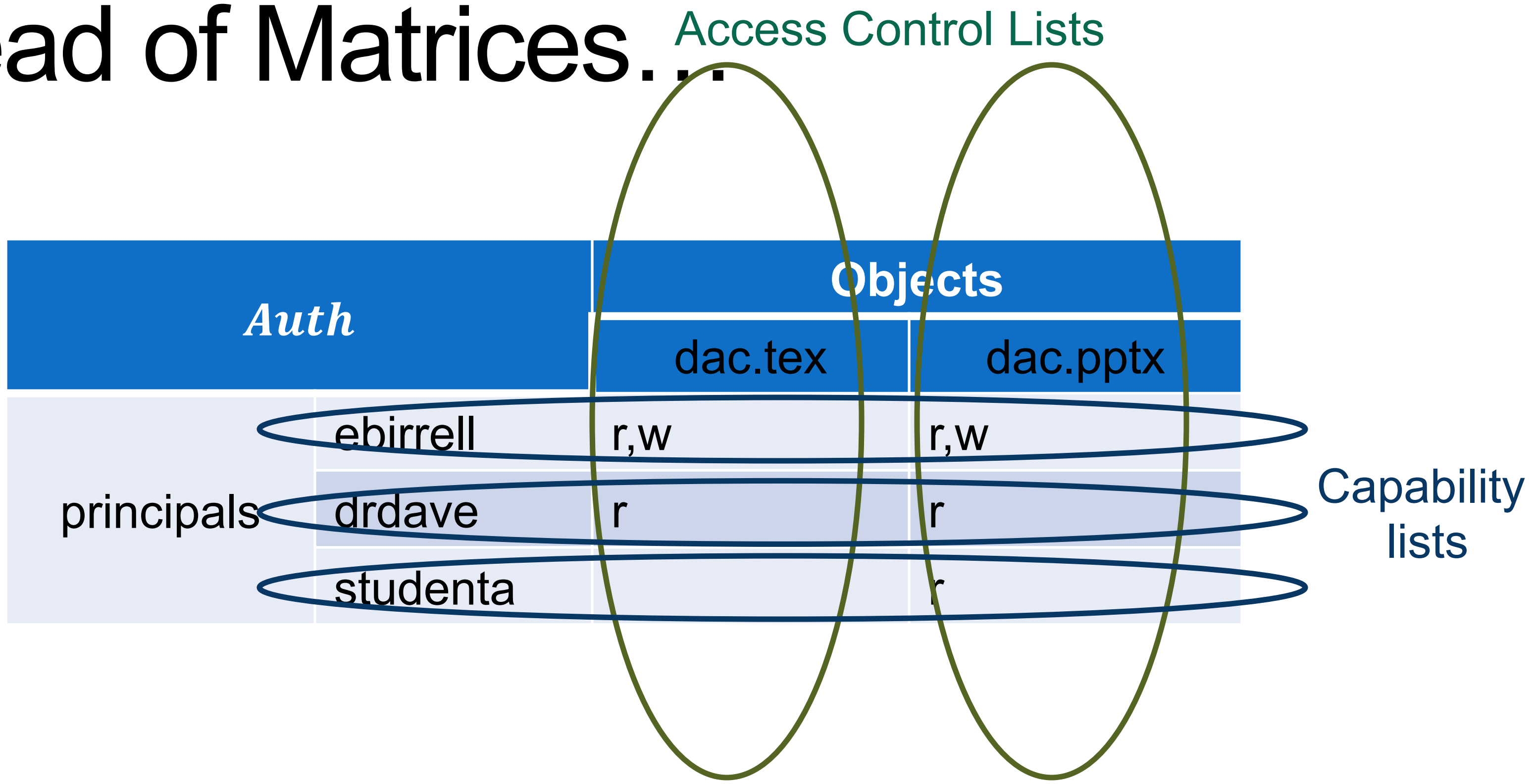
Exercise 1: Real-World Examples

- Consider two real-world access control systems: (i) **guest lists** at clubs, and (ii) **physical keys** to doors.
- How do each of those systems handle the primary concerns of access control:
  - granting access
  - preventing/determining access
  - revoking access
  - auditing access

Implementing DAC

- Need some way to representing authorization relation (matrix) $Auth$.
- That scheme must support certain functionality:
  - computing whether $\langle P, O, op \rangle \in Auth$ holds and (i.e., whether principal P is authorized to perform operation op on object O,
  - changing $Auth$ in accordance with defined commands
  - associating a protection domain with each thread of control
  - performing transitions between protection domains as execution proceeds.

Instead of Matrices…



Auth		Objects	
		dac.tex	dac.pptx
principals	ebirrell	r,w	r,w
	drdave	r	r
	studenta		r

- An **access control list** encodes the non-empty cells associated with a column (object).
- A **capability list** encode the non-empty cells associated with a row (principal).

Access Control Lists

- The access control list for an object O is a list
  $$\langle P_1, Privs_1\rangle, \langle P_2, Privs_2\rangle, \ldots\ , \langle P_n, Privs_n\rangle$$
  - e.g., ⟨ebirrell, {r,w}⟩ ⟨drdave, {r}⟩ ⟨studenta, {r}⟩
- To check whether P_i is allowed to perform op on object O,
  - Look up P_i in ACL. If not in list, reject op.
  - Check whether op is in the sent $Privs_i$. If not , reject op.

Access Control Lists

- Advantages:
  - Efficient review of permissions for an object
  - Centralized enforcement is simple to deploy, verify
  - Revocation is straightforward

- Disadvantages:
  - Inefficient review of permissions for a principal
  - Large lists impede performance
  - Vulnerable to confused deputy attack

Groups in ACLs

- A group declaration associates a group name with a set of principals.
- The set is specified either by enumerating its elements or by giving a predicate that all principals in the set must satisfy.
- An ACL entry $\langle G, Privs \rangle$, where G is a group name and $Privs$ is a set of privileges, grants all privileges in $Privs$ to all principals P that are members of G.

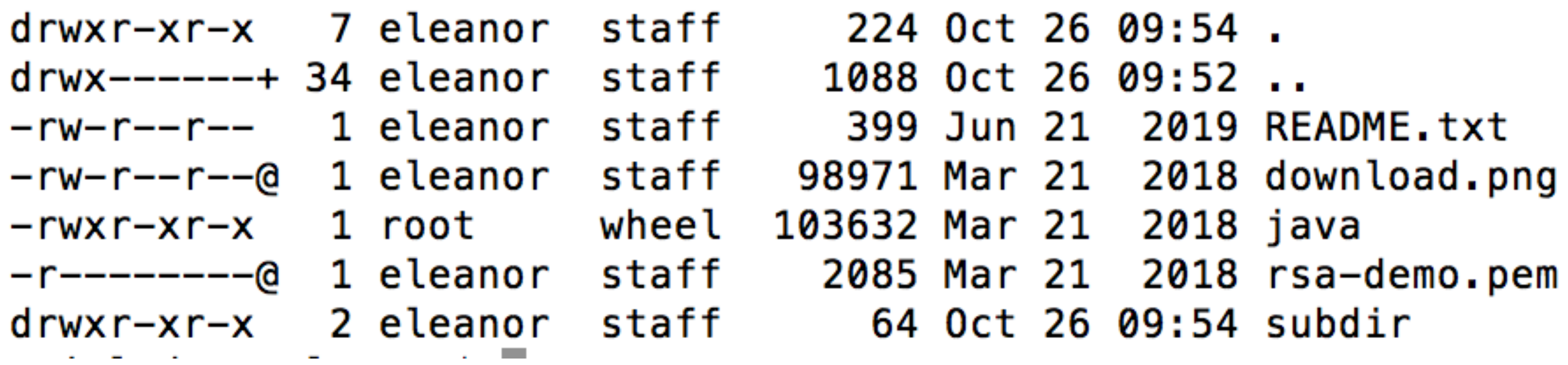
Wildcards

- Many advocate terse representations for ACL entries, assuming that checking shorter access control lists is faster.
- One approach is to employ patterns and wildcard symbols for specifying names of principals or privileges, so that a single ACL entry can replace many

Prohibitions

- In order to conclude that P does not hold op for an object O, we would have to enumerate and check the entire ACL.
- Some systems allow a prohibition to appear in an ACL-entry.
  - The prohibition $\overline{op}$ specifies that execution of operation op is prohibited.
  - Conflict resolution is not always specified (often first)

Demo: POSIX Access Control Lists



```
drwxr-xr-x    7 eleanor  staff     224 Oct 26 09:54 .
drwx------+ 34 eleanor  staff    1088 Oct 26 09:52 ..
-rw-r--r--    1 eleanor  staff     399 Jun 21  2019 README.txt
-rw-r--r--@  1 eleanor  staff   98971 Mar 21  2018 download.png
-rwxr-xr-x    1 root     wheel  103632 Mar 21  2018 java
-r--------@  1 eleanor  staff    2085 Mar 21  2018 rsa-demo.pem
drwxr-xr-x    2 eleanor  staff      64 Oct 26 09:54 subdir
```

Exercise 2: POSIX ACLs

- Consider a directory of your choice on your local machine and inspect the POSIX ACLs for the entries of that directory. Who is allowed to do what? Do these permissions satisfy the principle of least privilege?

- Consider the /data directory on the course vm and inspect the POSIX ACLs for the entries of that directory. What are you allowed to do? Do these permissions satisfy the principle of least privilege?

Protection Domains

- Motivation: users are too coarse-grained to define privileges
- **Protection Domains**:
  - Each thread of control is associated with a protection domain
  - Each protection domain is associated with a different set of privileges
  - We allow transitions from one protection domain to another as execution of the thread proceeds.

Protection Domains

- Typical implementation: certain system calls cause protection-domain transitions.
  - System calls for invoking a program or changing from user mode to supervisor mode are obvious candidates.
- Some operating systems provide an explicit domain-change system call instead
  - the application programmer or a compiler’s code generator is then required to decide when to invoke this domain-change system call
- We use the term attenuation of privilege for a transition into a protection domain that eliminates privileges.
- We use the term amplification of privilege for a transition into a protection domain that adds privileges.

Protection Domains

		Objects				
		dac.tex	dac.pptx	ebirrell @sh	ebirrell @edit	ebirrell@ powerpoint
principals	ebirrell@sh			e	e	e
	ebirrell@edit	r,w				
	ebirrell@powerpoint		r,w			
	drdave@sh					
	drdave@edit	r				
	drdave@powerpoint		r			
	studenta@sh					
	studenta@edit					
	studenta@powerpoint		r			

Role-Based Access Control

- Particularly in corporate and institutional settings, users might be granted privileges by virtue of membership in a group.
  - E.g., students who enroll in a class should be given access to that semester's class notes and assignments simply due to their new **role**
- Without groups, implementing role-based access control is error prone
  - Adding or deleting a member might require updating many access control lists. That can be error-prone.
  - Revocation is subtle. Should permission be removed with principal is removed from a group?

Exercise 3: RBAC

- What roles might you want to include in a course management system?

Exercise 4: Feedback

1. Rate how well you think this recorded lecture worked
   1. Better than an in-person class
   2. About as well as an in-person class
   3. Less well than an in-person class, but you still learned something
   4. Total waste of time, you didn't learn anything

2. How much time did you spend on this video lecture (including time spent on exercises)?

3. Do you have particular questions you would like me to address class?

4. Do you have any other comments or feedback?