Lecture notes for 4.3-4.4 access control

Access control
- Determination of which subjects can access which objects with what type of access
  - Tuple: (S, O, P)
- Policy: Description of the above as rules
- Enforcement: Constraint of access to policy
- Complete mediation: all ways of accessing object must have enforcement checks

Ways of implementing access control (to make this concrete, think in terms of file access)
- (how to manage tuples)
- User directory
- File access control list
- Capability

Example: 3 subjects ABC, 3 objects XYZ
(A, X, rwx)
(A, Y, r)
(B, Y, rw)
(B, Z, rx)
(C, X, rx)
(C, Y, rw)
(C, Z, rx)
Use this example for the below

Directory
- For each user, keep list of objects, permissions
- Lists can become large if many files world accessible
- No real-world example

ACL
- For each file, keep list of users & permissions
- Most real filesystems use this
- UNIX semantics of owner, group, world (suid will be later today)
- Windows semantics, inheritance from parent directories

Access control matrix
- Subjects down side
- Objects across top
- access permissions in the interior
- Matrix is sparse, may not be efficient representation

Capabilities
- Subjects carry tokens/tickets giving capability to access object
- Enforcer allows access if presented token along with access request
- Some component issued capability to suitable subject
- Tickets often protected with crypto
- Similar in idea to Kerberos (later in the semester)

Mandatory access control
- Policy sent centrally by security administrator
- users cannot influence policy
- Example: SELinux

Discretionary access control
- Users make policy decisions
- Set policy attributes on owned objects
- Example: File systems in commodity operating systems

Role-based access control
- Level of access depends upon current role
- Select role when logging in, view depends on choice
- logout & login again with different role to see different view

UNIX setuid / Windows runas
- Normal execution runs program as user executing
- Setuid execution runs program as owner of executable file
- Used when access beyond invoking user is required
- Example: UNIX su program
- Example: file storing high scores in a game... do not want users to edit directly via a text editor... use setuid on the game that matches subject with write access to the file