

CIS/TCOM 551

# Computer and Network Security

Spring 2005

Lecture 13

# Announcements

---

- Final project is on the web site.
  - Check web site for updates
  - Ask questions on the newsgroup (and check there for answers)
  - Due last day of classes: Friday, April 22nd.
- Final exam:
  - Thurs. April 28th 11am - 1pm
  - Chemistry B13

# Outline

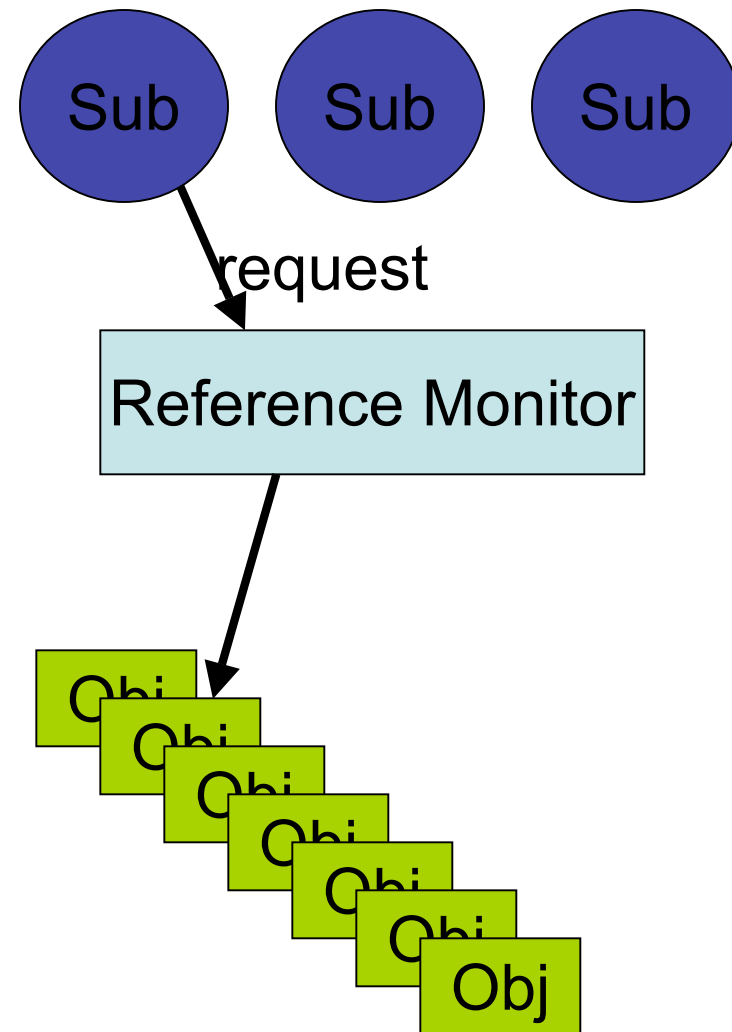
---

- Access Control Concepts
  - Matrix, ACL, Capabilities
- OS Mechanisms
  - Multics
    - Ring structure
  - Amoeba
    - Distributed, capabilities
  - Unix
    - File system, Setuid
  - Windows
    - File system, Tokens, EFS
  - SE Linux
    - Role-based, Domain type enforcement
- Secure OS
  - Methods for resisting stronger attacks
- Assurance
  - Multi-level security (MLS)
  - Orange Book, TCSEC
  - Common Criteria
  - Windows 2000 certification
- Some slides courtesy of John Mitchell

# Access Control: The Big Picture

---

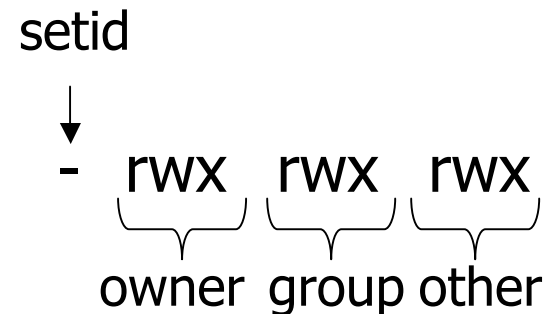
- Objects - resources being protected
  - E.g. files, devices, etc.
- Subjects - active entities
  - E.g. processes, machines
- Permissions (or Rights) - Kinds of access requests that are monitored
  - E.g. read, write, execute
- Reference monitor - mediates requests made by subjects
  - Permits or denies access



# Unix file security

---

- Each file has owner and group
- Permissions set by owner
  - Read, write, execute
  - Owner, group, other
  - Represented by vector of four octal values
- Only owner, root can change permissions
  - This privilege cannot be delegated or shared
- Setid bits – Discuss in a few slides



# Question

---

- "owner" can have fewer privileges than "other"
  - What happens?
    - User gets access?
    - User does not?
  
- Prioritized resolution of differences
  - if user = owner then *owner* permission
  - else if user in group then *group* permission
  - else *other* permission

# Effective User ID (EUID)

---

- Each process has three Ids (+ more under Linux)
  - Real user ID (RUID)
    - same as the user ID of parent (unless changed)
    - used to determine which user started the process
  - Effective user ID (EUID)
    - from set user ID bit on the file being executed, or sys call
    - determines the permissions for process
      - file access and port binding
  - Saved user ID (SUID)
    - So previous EUID can be restored
- Real group ID, effective group ID, used similarly

# Process Operations and IDs

---

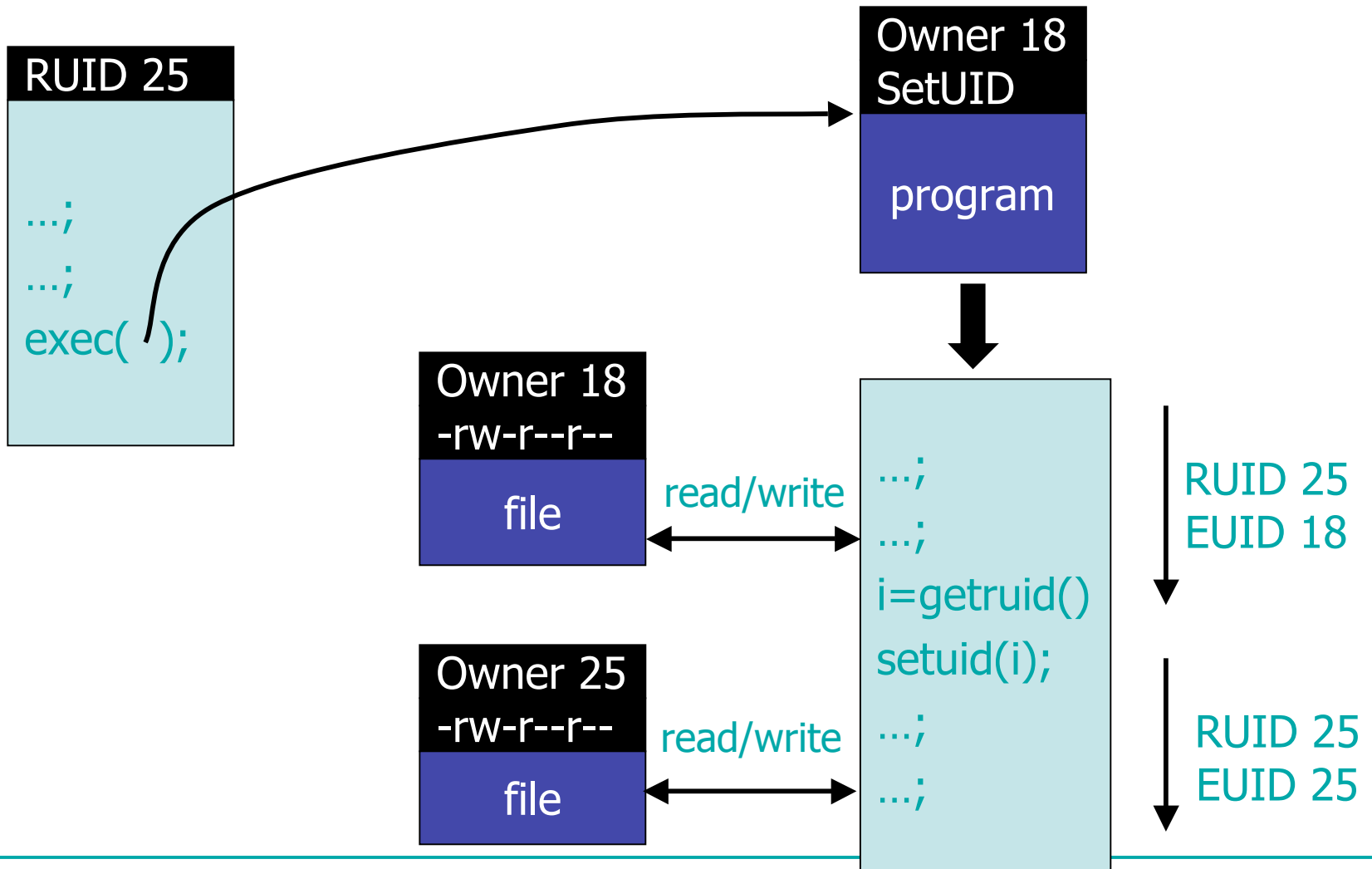
- Root
  - ID=0 for superuser root; can access any file
- Fork and Exec
  - Inherit three IDs, except exec of file with setuid bit
- Setuid system calls
  - seteuid(newid) can set EUID to
    - Real ID or saved ID, regardless of current EUID
    - Any ID, if EUID=0
- Details are actually more complicated
  - Several different calls: setuid, seteuid, setreuid

# Setid bits on executable Unix file

---

- Three setid bits
  - Setuid – set EUID of process to ID of file owner
  - Setgid – set EGID of process to GID of file
  - Sticky
    - Off: if user has write permission on directory, can rename or remove files, even if not owner
    - On: only file owner, directory owner, and root can rename or remove file in the directory

# Example



# Setuid programming

---

- Can do anything that owner of file is allowed to do
- Be Careful!
  - Root can do anything; don't get tricked (no middle ground)
  - Principle of least privilege – change EUID when root privileges no longer needed
  - Be sure not to
    - Take action for untrusted user
    - Return secret data to untrusted user
- Setuid scripts
  - This is a bad idea
  - Historically, race conditions
    - Begin executing setuid program; change contents of program before it loads and is executed

# Unix summary

---

- We're all very used to this ...
  - So probably seems pretty good
  - We overlook ways it might be better
- Good things
  - Some protection from most users
  - Flexible enough to make things possible
- Main bad thing
  - Too tempting to use root privileges
  - No way to assume some root privileges without all root privileges

# Access control in Windows (NTFS)

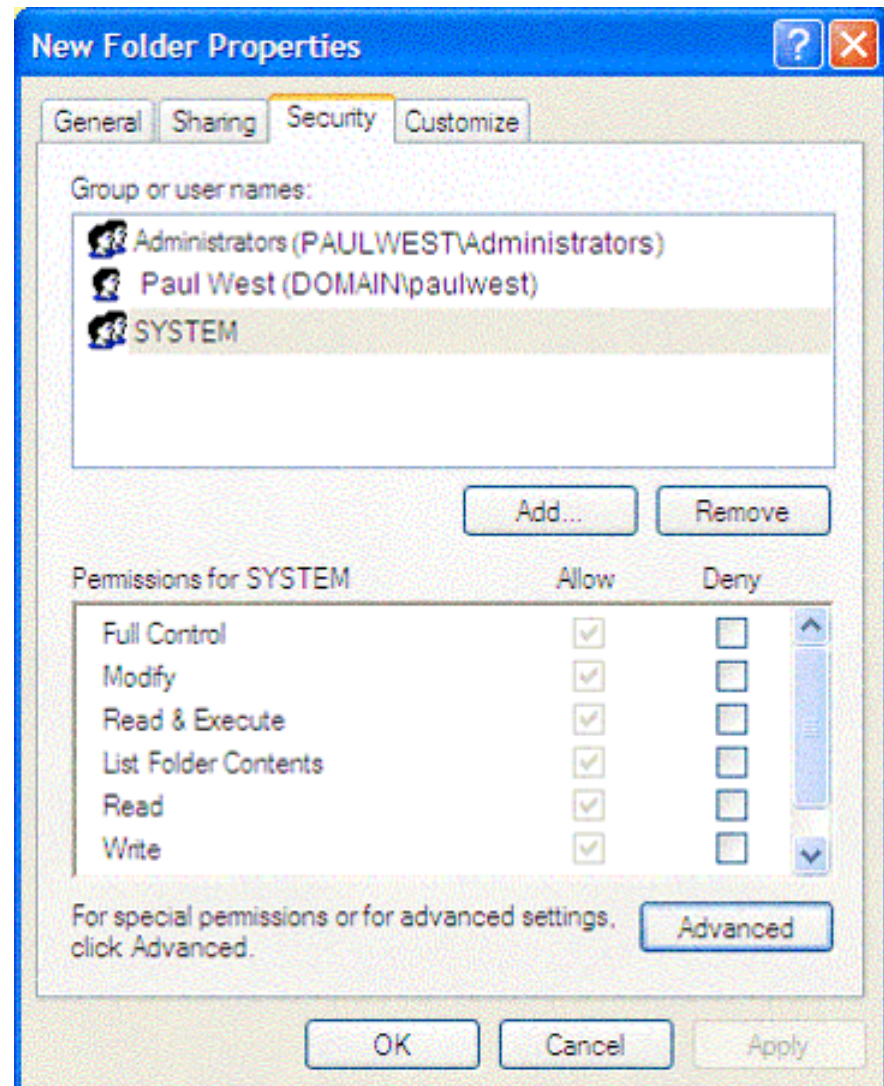
---

---

- Some basic functionality similar to Unix
  - Specify access for groups and users
    - Read, modify, change owner, delete
- Some additional concepts
  - Tokens
  - Security attributes
- Generally
  - More flexibility than Unix
    - Can define new permissions
    - Can give some but not all administrator privileges

# Sample permission options

- SID
  - Identity (replaces UID)
    - SID revision number
    - 48-bit authority value
    - variable number of Relative Identifiers (RIDs), for uniqueness
  - Users, groups, computers, domains, domain members all have SIDs



# Permission Inheritance

---

- Static permission inheritance (Win NT)
  - Initially, subfolders inherit permissions of folder
  - Folder, subfolder changed independently
  - *Replace Permissions on Subdirectories* command
    - Eliminates any differences in permissions
- Dynamic permission inheritance (Win 2000)
  - Child inherits parent permission, remains linked
  - Parent changes are inherited, except explicit settings
  - Inherited and explicitly-set permissions may conflict
    - Resolution rules
      - Positive permissions are additive (take union of all permissions)
      - Negative permission (deny access) takes priority

# Tokens

---

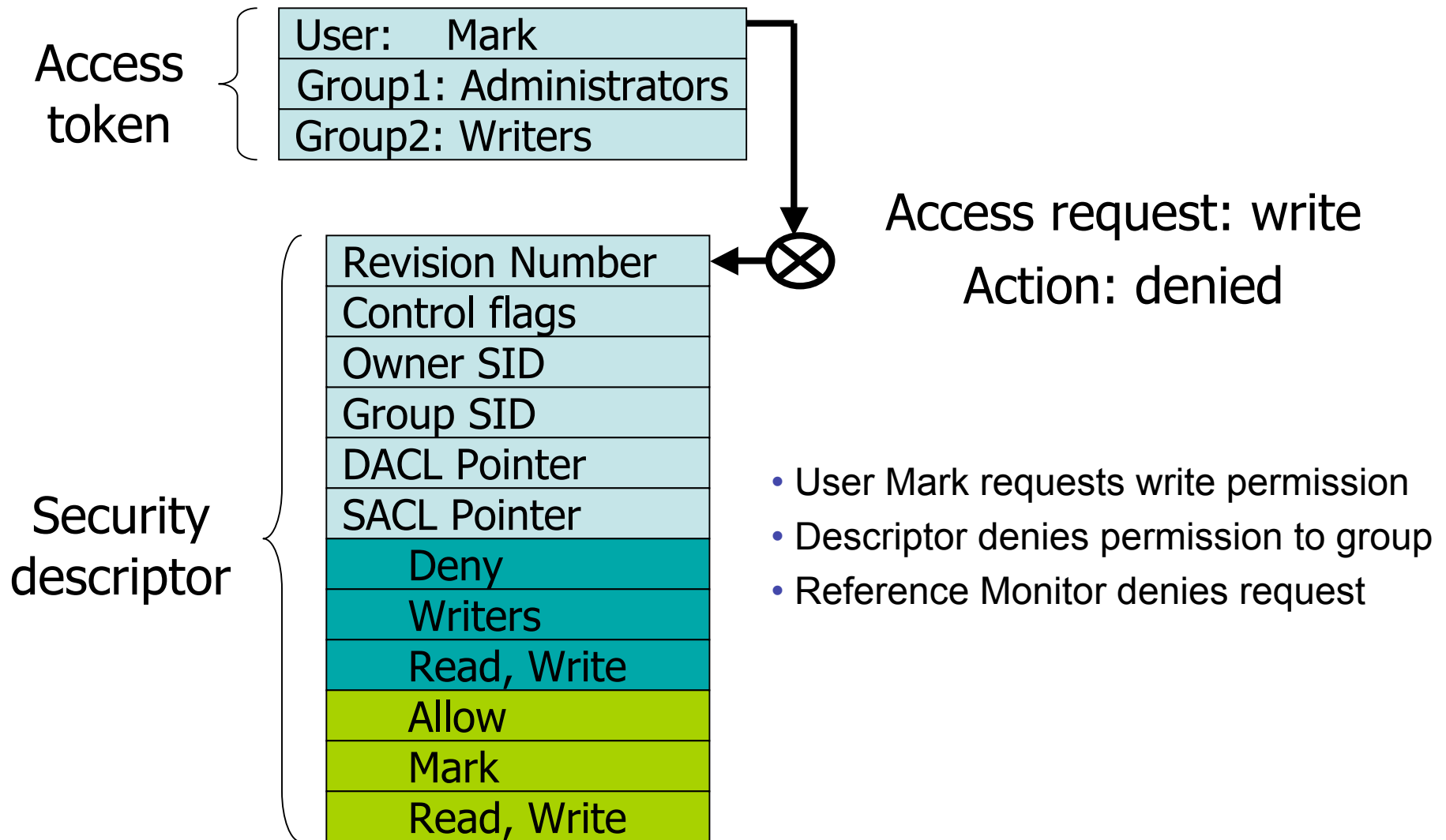
- Security Reference Monitor
  - uses tokens to identify the security context of a process or thread
- Security context
  - privileges, accounts, and groups associated with the process or thread
- Impersonation token
  - thread uses temporarily to adopt a different security context, usually of another user
  
- Related to the EUID used in Unix.

# Security Descriptor

---

- Information associated with an object
  - who can perform what actions on the object
- Several fields
  - Header
    - Descriptor revision number
    - Control flags, attributes of the descriptor
      - E.g., memory layout of the descriptor
  - SID of the object's owner
  - SID of the primary group of the object
  - Two attached optional lists:
    - Discretionary Access Control List (DACL) – users, groups, ...
    - System Access Control List (SACL) – system logs, ..

# Example access request



# Impersonation Tokens

---

- Windows equivalent of setuid
- Process uses security attributes of another
  - Client passes impersonation token to server
- Client specifies impersonation level of server
  - Anonymous
    - Token has no information about the client
  - Identification
    - server obtain the SIDs of client and client's privileges, but server cannot impersonate the client
  - Impersonation
    - server identify and impersonate the client
  - Delegation
    - lets server impersonate client on local, remote systems