Access Control and Operating System Security

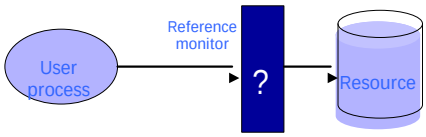
John Mitchell

Outline

- Access Control
  - Matrix, ACL, Capabilities
  - Multi-level security (MLS)
- OS Policies
  - Multics
    - Ring structure
  - Unix
    - File system, Setuid
  - Windows
    - File system, Tokens, EFS
  - SE Linux
    - Role-based
    - Domain type enforcement
- Secure OS
  - Methods for resisting stronger attacks
- Assurance
  - Orange Book, TCSEC
  - Common Criteria
  - Windows 2000 certification
- Some Limitations
  - Information flow
  - Covert channels

Access control

- Common Assumption
  - System knows who the user is
    - User has entered a name and password, or other info
  - Access requests pass through gatekeeper
    - Global property; OS must be designed so that this is true



Decide whether user can apply operation to resource

Access control matrix [Lampson]

	File 1	File 2	File 3	...	File n
User 1	read	write	-	-	read
User 2	write	write	write	-	-
User 3	-	-	-	read	read
...					
User m	read	write	read	write	read

Two implementation concepts

- Access control list (ACL)
  - Store column of matrix with the resource
- Capability
  - Allow user to hold a "ticket" for each resource
  - Roughly: store row of matrix with the user

	File 1	File 2	...
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
...			
User m	read	write	write

Access control lists are widely used, often with groups

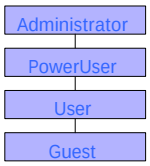
Some aspects of capability concept are used in Kerberos, ...

Capabilities

- Operating system concept
  - "... of the future and always will be ..."
- Examples
  - Dennis and van Horn, MIT PDP-1 Timesharing
  - Hydra, StarOS, Intel iAPX 432, Amoeba, Eros, ...
- Reference
  - Henry Levy, Capability-based Computer Systems
    http://www.cs.washington.edu/homes/levy/capabook/

Roles (also called Groups)

- Role = set of users
  - Administrator, PowerUser, User, Guest
  - Assign permissions to roles; each user gets permission
- Role hierarchy
  - Partial order of roles
  - Each role gets permissions of roles below
  - List only new permissions given to each role



Groups for resources, rights

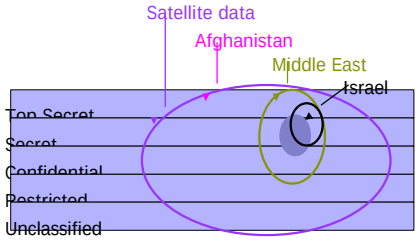
- Permission = ⟨right, resource⟩
- Group related resources
- Hierarchy for rights or resources
  - If user has right r, and r>s, then user has right s
  - If user has read access to directory, user has read access to every file in directory
- Big problem in access control
  - Complex mechanisms require complex input
  - Difficult to configure and maintain
  - Roles, other organizing ideas try to simplify problem

Multi-level Security Concepts

- Military security policy
  - Classification involves sensitivity levels, compartments
  - Do not let classified information leak to unclassified files
- Group individuals and resources
  - Use some form of hierarchy to organize policy
- Other concepts
  - Separation of duty
  - Chinese Wall Policy

Military security policy

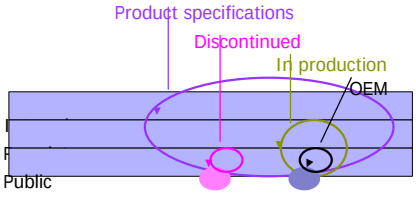
- Sensitivity levels
- Compartments



Military security policy

- Classification of personnel and data
  - Class = ⟨rank, compartment⟩
- Dominance relation
  - $D_1 \leq D_2$ iff $rank_1 \leq rank_2$ and $compartment_1 \subseteq compartment_2$
  - Example: ⟨Restricted, Israel⟩ ≤ ⟨Secret, Middle East⟩
- Applies to
  - Subjects – users or processes
  - Objects – documents or resources

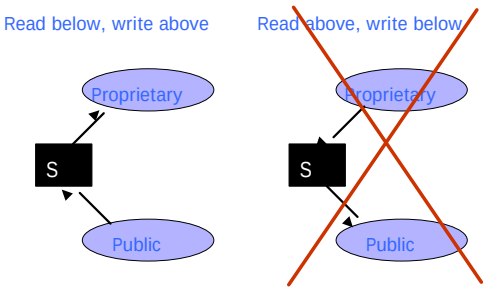
Commercial version



Bell-LaPadula Confidentiality Model

- When is it OK to release information?
- Two Properties (with silly names)
  - Simple security property
    - A subject S may read object O only if C(O) ≤ C(S)
  - *-Property
    - A subject S with read access to O may write object P only if C(O) ≤ C(P)
- In words,
  - You may only *read below* your classification and only *write above* your classification

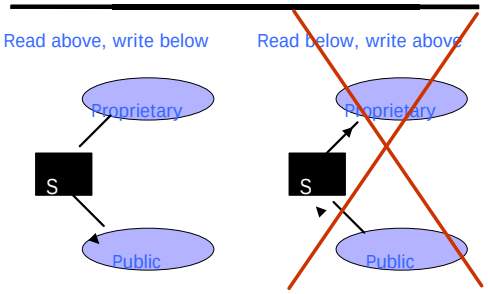
Picture: Confidentiality



Biba Integrity Model

- Rules that preserve integrity of information
- Two Properties (with silly names)
  - Simple integrity property
    - A subject S may write object O only if C(S) ≥ C(O)
    (Only trust S to modify O if S has higher rank ...)
  - *-Property
    - A subject S with read access to O may write object P only if C(O) ≥ C(P)
    (Only move info from O to P if O is more trusted than P)
- In words,
  - You may only *write below* your classification and only *read above* your classification

Picture: Integrity



Problem: Models are contradictory

- Bell-LaPadula Confidentiality
  - Read down, write up
- Biba Integrity
  - Read up, write down
- Want both confidentiality and integrity
  - Only way to satisfy both models is only allow read and write at same classification

In reality: Bell-LaPadula used more than Biba model
Example: Common Criteria

Other policy concepts

- Separation of duty
  - If amount is over $10,000, check is only valid if signed by two authorized people
  - Two people must be *different*
  - Policy involves role membership and ≠
- Chinese Wall Policy
  - Lawyers L1, L2 in Firm F are experts in banking
  - If bank B1 sues bank B2,
    - L1 and L2 can each work for either B1 or B2
    - No lawyer can work for opposite sides in any case
  - Permission depends on use of other permissions

Example OS Mechanisms

- Multics
- Unix
- Windows
- SE Linux (briefly)

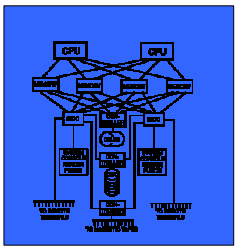
Multics

- Operating System
  - Designed 1964-1967
    - MIT Project MAC, Bell Labs, GE
  - At peak, ~100 Multics sites
  - Last system, Canadian Department of Defense, Nova Scotia, shut down October, 2000
- Extensive Security Mechanisms
  - Influenced many subsequent systems

http://www.multicians.org/security.html

Multics time period

- Timesharing was new concept
  - Serve Boston area with one 386-based PC



Multics Innovations

- Segmented, Virtual memory
  - Hardware translates virtual address to real address
- High-level language implementation
  - Written in PL/1, only small part in assembly lang
- Shared memory multiprocessor
  - Multiple CPUs share same physical memory
- Relational database
  - Multics Relational Data Store (MRDS) in 1978
- Security
  - Designed to be secure from the beginning
  - First B2 security rating (1980s), only one for years

Multics Access Model

- Ring structure
  - A ring is a domain in which a process executes
  - Numbered 0, 1, 2, ... ; Kernel is ring 0
  - Graduated privileges
    - Processes at ring i have privileges of every ring j > i
- Segments
  - Each data area or procedure is called a segment
  - Segment protection ⟨b1, b2, b3⟩ with b1 ≤ b2 ≤ b3
    - Process/data can be accessed from rings b1 ... b2
    - A process from rings b2 ... b3 can only call segment at restricted entry points

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

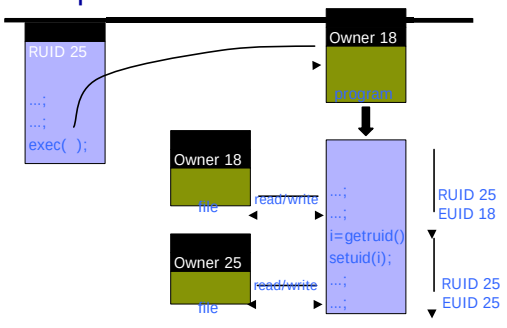
Process Operations and IDs

- 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

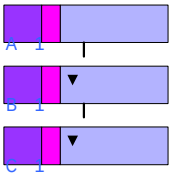
- Three setid bits
  - Setuid – set EUID of process to ID of file owner
  - Setgid – set EGID of process to GID of file
  - Sticky
    - Off: if user has write permission on directory, can rename or remove files, even if not owner
    - On: only file owner, directory owner, and root can rename or remove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

Example



Compare to stack inspection

- Careful with Setuid !
  - Can do anything that owner of file is allowed to do
  - Be sure not to
    - Take action for untrusted user
    - Return secret data to untrusted user



Note: anything possible if root; no middle ground between user and root

Setuid programming

- We talked about this before …
- Be Careful!
  - Root can do anything; don' t get tricked
  - Principle of least privilege – change EUID when root privileges no longer needed
- 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

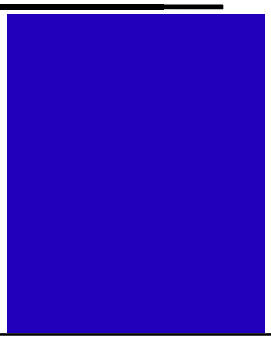
- We're all very used to this …
  - So probably seems pretty good
  - We overlook ways it might be better
- Good things
  - Some protection from most users
  - Flexible enough to make things possible
- Main bad thing
  - Too tempting to use root privileges
  - No way to assume some root privileges without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

Access control in Windows (NTFS)

- 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
      - Negative permission (deny access) takes priority

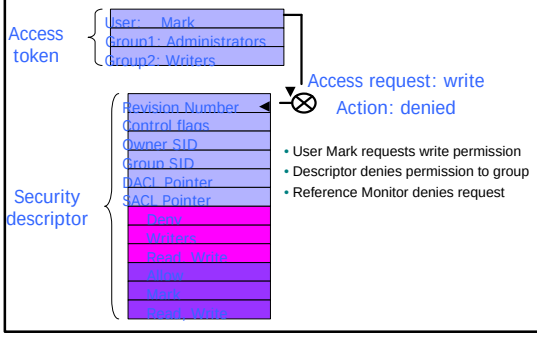
Tokens

- 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

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



- User Mark requests write permission
- Descriptor denies permission to group
- Reference Monitor denies request

Impersonation Tokens (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

Encrypted File Systems (EFS, CFS)

- Store files in encrypted form
  - Key management: user's key decrypts file
  - Useful protection if someone steals disk
- Windows – EFS
  - User marks a file for encryption
  - Unique file encryption key is created
  - Key is encrypted, can be stored on smart card
- Unix – CFS [Matt Blaze]
  - Transparent use
  - Local NFS server running on "loopback" interface
  - Key protected by passphrase

Q: Why use crypto file system?

- General security questions
  - What properties are provided?
  - Against what form of attack?
- Crypto file system
  - What properties?
    - Secrecy, integrity, authenticity, ... ?
  - Against what kinds of attack?
    - Someone steals your laptop?
    - Someone steals your removable disk?
    - Someone has network access to shared file system?

  Depends on how file system configured and used

SELinux Security Policy Abstractions

- Type enforcement
  - Each process has an associated domain
  - Each object has an associated type
  - Configuration files specify
    - How domains are allowed to access types
    - Allowable interactions and transitions between domains
- Role-based access control
  - Each process has an associated role
    - Separate system and user processes
  - configuration files specify
    - Set of domains that may be entered by each role

Secure Operating Systems

- Extra mechanisms for extra security
- Follow design and implementation procedures
- Review of design and implementation
- Maintenance procedures

Will discuss

- Mechanisms associated with secure OS
- Standards for certification
  - Mostly used by government, some commercial interest

Sample Features of Trusted OS

- Mandatory access control
  - MAC not under user control, precedence over DAC
- Object reuse protection
  - Write over old data when file space is allocated
- Complete mediation
  - Prevent any access that circumvents monitor
- Audit
  - See next slide
- Intrusion detection
  - Anomaly detection
    - Learn normal activity, Report abnormal actions
  - Attack detection
    - Recognize patterns associated with known attacks

Audit

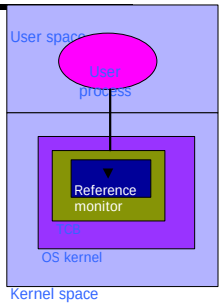
- Log security-related events
- Protect audit log
  - Write to write-once non-volatile medium
- Audit logs can become huge
  - Manage size by following policy
    - Storage becomes more feasible
    - Analysis more feasible since entries more meaningful
  - Example policies
    - Audit only first, last access by process to a file
    - Do not record routine, expected events
      - E.g., starting one process always loads ...

Trusted path

- Spoofing
  - Fool user/process into thinking they are communicating with secure part of system
  - Intercept communication
- Trusted path
  - Mechanisms to prevent spoofing
    - Special key sequence for passwd command intercepted by trusted kernel (e.g, ctrl-alt-delete)
    - Allow some actions only at boot time, before user processes loaded

Kernelized Design

- Trusted Computing Base
  - Hardware and software for enforcing security rules
- Reference monitor
  - Part of TCB
  - All system calls go through reference monitor for security checking
  - Most OS not designed this way



SELinux

- Security-enhanced Linux system (NSA)
  - Enforce separation of information based on confidentiality and integrity requirements
  - Mandatory access control incorporated into the major subsystems of the kernel
    - Limit tampering and bypassing of application security mechanisms
    - Confine damage caused by malicious applications

http://www.nsa.gov/selinux/

Why Linux?

- Open source
  - Already subject to public review
    - This by itself does not guarantee security ...
  - NSA can review source, modify and extend
  - Hope to encourage additional operating system security research
  - Released under the same terms and conditions as the original sources.
    - includes documentation and source code

Rainbow Series

DoD Trusted Computer Sys Evaluation Criteria (Orange Book)
Audit in Trusted Systems (Tan Book)
Configuration Management in Trusted Systems (Amber Book)
Trusted Distribution in Trusted Systems (Dark Lavender Book)
Security Modeling in Trusted Systems (Aqua Book)
Formal Verification Systems (Purple Book)
Covert Channel Analysis of Trusted Systems (Light Pink Book)
... many more

http://www.radium.ncsc.mil/tpep/library/rainbow/index.html

Assurance methods

- Testing
  - Can demonstrate existence of flaw, not absence
- Formal verification
  - Time-consuming, painstaking process
- "Validation"
  - Requirements checking
  - Design and code reviews
    - Sit around table, drink lots of coffee, ...
  - Module and system testing

Orange Book Criteria (TCSEC)

- Level D
  - No security requirements
- Level C For environments with cooperating users
  - C1 – protected mode OS, authenticated login, DAC, security testing and documentation (Unix)
  - C2 – DAC to level of individual user, object initialization, auditing (Windows NT 4.0)
- Level B, A
  - All users and objects must be assigned a security label (classified, unclassified, etc.)
  - System must enforce Bell-LaPadula model

Levels B, A (continued)

- Level B
  - B1 – classification and Bell-LaPadula
  - B2 – system designed in top-down modular way, must be possible to verify, covert channels must be analyzed
  - B3 – ACLs with users and groups, formal TCB must be presented, adequate security auditing, secure crash recovery
- Level A1
  - Formal proof of protection system, formal proof that model is correct, demonstration that impl conforms to model, formal covert channel analysis

Orange Book Requirements (TCSEC)

- Security Policy
- Accountability
- Assurance
- Documentation

- Next few slides: details not important ...
  - Main point: Higher levels require more work ..., documentation and configuration management are part of the criteria

Common Criteria

- Three parts
  - CC Documents
    - Protection profiles: requirements for category of systems
      - Functional requirements
      - Assurance requirements
  - CC Evaluation Methodology
  - National Schemes (local ways of doing evaluation)
- Endorsed by 14 countries
- Replaces TCSEC
  - CC adopted 1998
  - Last TCSEC evaluation completed 2000

http://www.commoncriteria.org/

Protection Profiles

- Requirements for categories of systems
  - Subject to review and certified
- Example: Controlled Access PP (CAPP_V1.d)
  - Security functional requirements
    - Authentication, User Data Protection, Prevent Audit Loss
  - Security assurance requirements
    - Security testing, Admin guidance, Life-cycle support, …
  - Assumes non-hostile and well-managed users
  - Does not consider malicious system developers

Evaluation Assurance Levels 1 – 4

EAL 1: Functionally Tested
- Review of functional and interface specifications
- Some independent testing

EAL 2: Structurally Tested
- Analysis of security functions, incl high-level design
- Independent testing, review of developer testing

EAL 3: Methodically Tested and Checked
- Development environment controls; config mgmt

EAL 4: Methodically Designed, Tested, Reviewed
- Informal spec of security policy, Independent testing

Evaluation Assurance Levels 5 – 7

EAL 5: Semiformally Designed and Tested
- Formal model, modular design
- Vulnerability search, covert channel analysis

EAL 6: Semiformally Verified Design and Tested
- Structured development process

- EAL 7: Formally Verified Design and Tested
  - Formal presentation of functional specification
  - Product or system design must be simple
  - Independent confirmation of developer tests

Example: Windows 2000, EAL 4+

- Evaluation performed by SAIC
- Used “Controlled Access Protection Profile”
- Level EAL 4 + Flaw Remediation
  - “EAL 4 … represents the highest level at which products not built specifically to meet the requirements of EAL 5-7 ought to be evaluated.” (EAL 5-7 requires more stringent design and development procedures …)
  - Flaw Remediation
- Evaluation based on specific configurations
  - Produced configuration guide that may be useful

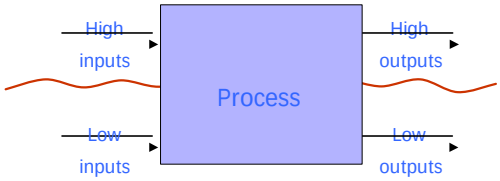
Is Windows is “Secure”?

- Good things
  - Design goals include security goals
  - Independent review, configuration guidelines
- But ...
  - “Secure” is a complex concept
    - What properties protected against what attacks?
  - Typical installation includes more than just OS
    - Many problems arise from applications, device drivers
    - Windows driver certification program

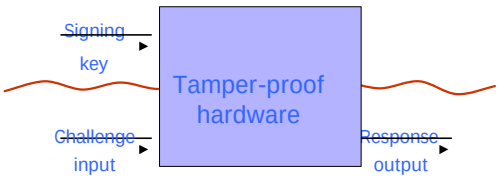
Limitations of Secure OS

- Noninterference
  - Actions by high-level users (secret, top secret) should not be observable by low-level users (unclassified, ...)
  - Difficult to achieve and prove, not impossible
- Covert Channels
  - Can user of system deliberately communicate secret information to external collaborator?

Noninterference



Example: Smart Card



Covert Channels

- Butler Lampson
  - Difficulty achieving confinement (paper on web)
  - Communicate by using CPU, locking/unlocking file, sending/delaying msg, ...
- Gustavus Simmons
  - Cryptographic techniques make it impossible to detect presence of a covert channel

Outline

- Access Control
  - Matrix, ACL, Capabilities
  - Multi-level security (MLS)
- OS Policies
  - Multics
    - Ring structure
  - Unix
    - File system, Setuid
  - Windows
    - File system, Tokens, EFS
  - SE Linux
    - Role-based
    - Domain type enforcement
- Secure OS
  - Methods for resisting stronger attacks
- Assurance
  - Orange Book, TCSEC
  - Common Criteria
  - Windows 2000 certification
- Some Limitations
  - Information flow
  - Covert channels