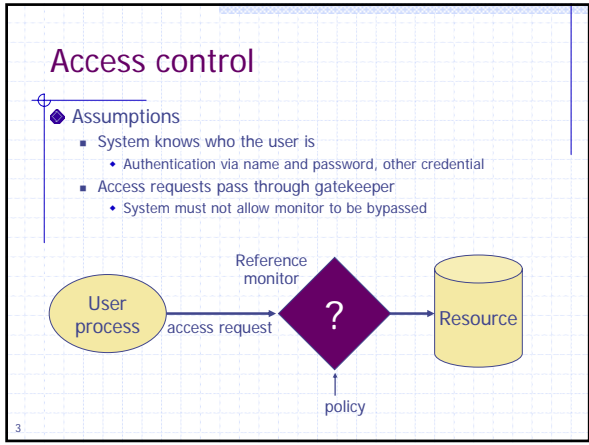


## Outline

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



## Access control matrix [Lampson]

		Objects				
		File 1	File 2	File 3	...	File n
Subjects	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
  - User holds a "ticket" for each resource
  - Two variations
    - store row of matrix with user, under OS control
    - unforgeable ticket in user space

	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

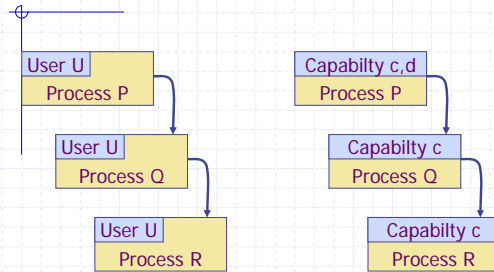
- ◆ Operating system concept
  - "... of the future and always will be ..."
- ◆ Examples
  - Dennis and van Horn, MIT PDP-1 Timesharing
  - Hydra, StarOS, Intel iAPX 432, Eros, ...
  - Amoeba: distributed, unforgeable tickets
- ◆ References
  - Henry Levy, Capability-based Computer Systems  
<http://www.cs.washington.edu/homes/levy/capabook/>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

## ACL vs Capabilities

- ◆ Access control list
  - Associate list with each object
  - Check user/group against list
  - Relies on authentication: need to know user
- ◆ Capabilities
  - Capability is unforgeable ticket
    - Random bit sequence, or managed by OS
    - Can be passed from one process to another
  - Reference monitor checks ticket
    - Does not need to know identity of user/process

7

## ACL vs Capabilities



8

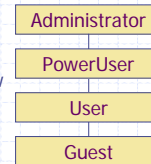
## ACL vs Capabilities

- ◆ Delegation
  - Cap: Process can pass capability at run time
  - ACL: Try to get owner to add permission to list?
    - More common: let other process act under current user
- ◆ Revocation
  - ACL: Remove user or group from list
  - Cap: Try to get capability back from process?
    - Possible in some systems if appropriate bookkeeping
      - OS knows what data is capability
      - If capability is used for multiple resources, have to revoke all or none ...
      - Other details ...

9

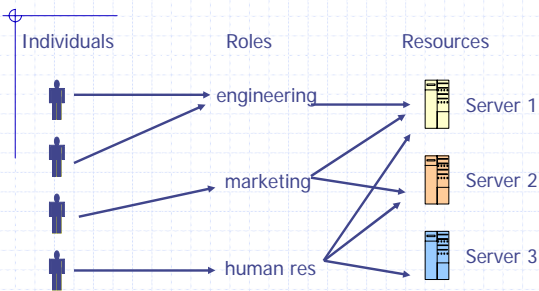
## Roles (also called Groups)

- ◆ 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



10

## Role-Based Access Control



Advantage: user's change more frequently than roles

11

## Groups for resources, rights

- ◆ Permission =  $\langle \text{right}, \text{resource} \rangle$
- ◆ Permission hierarchies
  - 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
- ◆ General problem in access control
  - Complex mechanisms require complex input
  - Difficult to configure and maintain
  - Roles, other organizing ideas try to simplify problem

12

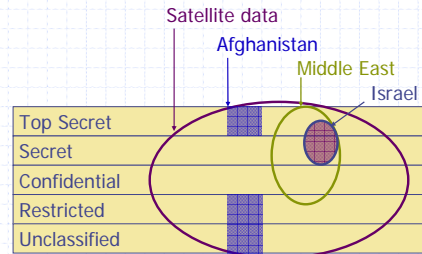
## Multi-Level Security (MLS) Concepts

- ◆ 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 policy concepts
  - Separation of duty
  - "Chinese Wall" Policy

13

## Military security policy

- ◆ Sensitivity levels
- ◆ Compartments



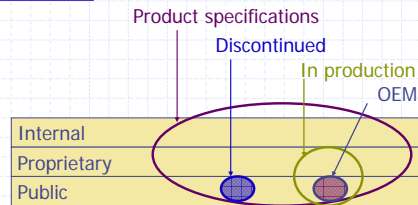
14

## 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)  $\leq$  (Secret, Middle East)
- ◆ Applies to
  - Subjects – users or processes
  - Objects – documents or resources

15

## Commercial version



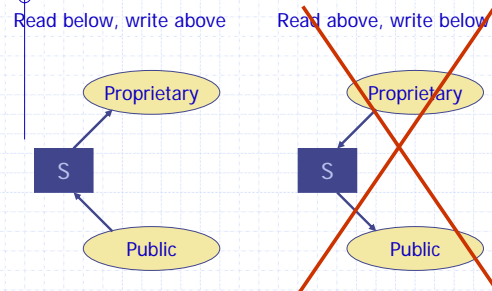
16

## 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  $C(O) \leq C(S)$
  - \*-Property
    - A subject S with read access to O may write object P only if  $C(O) \leq C(P)$
- ◆ In words,
  - You may only *read below* your classification and only *write above* your classification

17

## Picture: Confidentiality



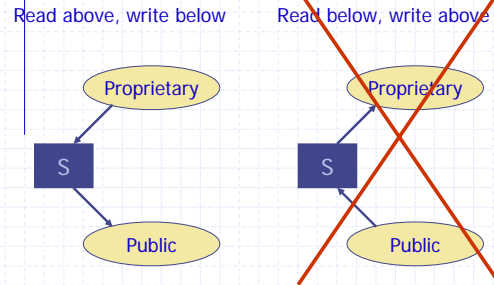
18

## 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  $C(S) \geq C(O)$   
(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  $C(O) \geq C(P)$   
(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

19

## Picture: Integrity



20

## Problem: Models appear contradictory

- ◆ Bell-LaPadula Confidentiality
  - Read down, write up
- ◆ Biba Integrity
  - Read up, write down
- ◆ Want both confidentiality and integrity
  - Contradiction is partly an illusion
  - May use Bell-LaPadula for some classification of personnel and data, Biba for another
    - ◆ Otherwise, only way to satisfy both models is only allow read and write at same classification

In reality: Bell-LaPadula used more than Biba model, e.g., Common Criteria

21

## 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  $\neq$
  - ◆ 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
- These policies cannot be represented using access matrix

22

## Example OS Mechanisms

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

23

## 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>

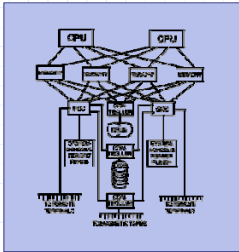
E.H. Organick, *The Multics System: An Examination of Its Structure*, MIT Press, 1972

## Multics time period



F.J. Corbato

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



25

## Multics Innovations

- ◆ Segmented, Virtual memory
  - Hardware translates virtual address to real address
- ◆ High-level language implementation
  - Written in PL/I, 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

26

## 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  $\langle b1, b2, b3 \rangle$  with  $b1 \leq b2 \leq b3$ 
    - ◆ Process/data can be accessed from rings  $b1 \dots b2$
    - ◆ A process from rings  $b2 \dots b3$  can only call segment at restricted entry points

27

## Multics process

- ◆ Multiple segments
  - Segments are dynamically linked
  - Linking process uses file system to find segment
  - A segment may be shared by several processes
- ◆ Multiple rings
  - Procedure, data segments each in specific ring
  - Access depends on two mechanisms
    - ◆ Per-Segment Access Control
      - File author specifies the users that have access to it
    - ◆ Concentric Rings of Protection
      - Call or read/write segments in outer rings
      - To access inner ring, go through a "gatekeeper"
- ◆ Interprocess communication through "channels"

28

## Amoeba

Server port	Obj #	Rights	Check field
-------------	-------	--------	-------------

- ◆ Distributed system
  - Multiple processors, connected by network
  - Process on A can start a new process on B
  - Location of processes designed to be transparent
- ◆ Capability-based system
  - Each object resides on server
  - Invoke operation through message to server
    - ◆ Send message with capability and parameters
    - ◆ Server uses object # to identify object
    - ◆ Server checks rights field to see if operation is allowed
    - ◆ Check field prevents processes from forging capabilities

29

## Capabilities

Server port	Obj #	Rights	Check field
-------------	-------	--------	-------------

- ◆ Owner capability
  - When server creates object, returns owner cap.
    - ◆ All rights bits are set to 1 (= allow operation)
    - ◆ Check field contains 48-bit rand number stored by server
- ◆ Derived capability
  - Owner can set some rights bits to 0
  - Calculate new check field
    - ◆ XOR rights field with random number from check field
    - ◆ Apply one-way function to calculate new check field
  - Server can verify rights and check field
    - ◆ Without owner capability, cannot forge derived capability

Protection by user-process at server; no special OS support needed

30

## 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



31

## Question

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

32

## 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

33

## 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

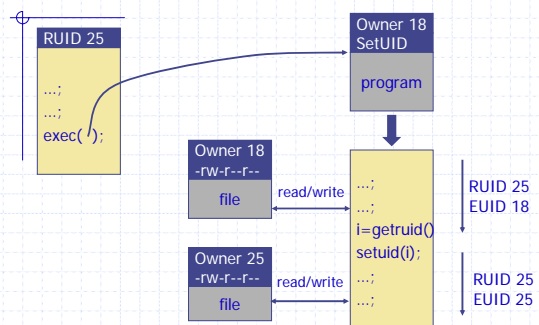
34

## 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

35

## Example

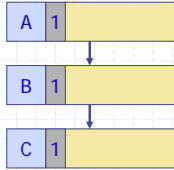


36

## 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

37

## 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

38

## Unix summary

### ◆ Many of you may be 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

39

## 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

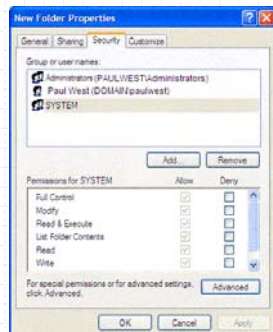
- More flexibility than Unix
  - Can define new permissions
  - Can give some but not all administrator privileges

40

## Sample permission options

### ◆ Security ID (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



41

## 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

42

## 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

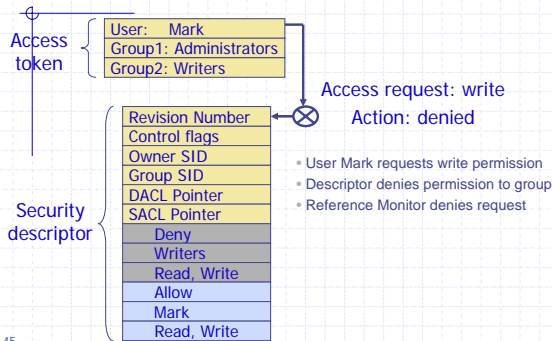
43

## 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, ...

44

## Example access request



45

## 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

46

## 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

47

## Outline

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

48



## What makes a "secure" OS?

- ◆ Extra security features (compared to ordinary OS)
  - Stronger authentication mechanisms
    - Example: require token + password
  - More security policy options
    - Example: only let users read file f for purpose p
  - Logging and other features
- ◆ More secure implementation
  - Apply secure design and coding principles
  - Assurance and certification
    - Code audit or formal verification
  - Maintenance procedures
    - Apply patches, etc.

49

## 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
  - Log security-related events and check logs
- ◆ Intrusion detection
  - Anomaly detection
    - Learn normal activity, Report abnormal actions
  - Attack detection
    - Recognize patterns associated with known attacks

50

## Controlling information flow

- ◆ MAC policy
  - Information from one object may only flow to an object at the same or at a higher security level
- ◆ Conservative approach
  - Information flow takes place when an object changes its state or when a new object is created
- ◆ Implementation as access policy
  - If a process reads a file at one security level, it cannot create or write a file at a lower level
  - This is *not* a DAC policy, not an ACL policy

51

## 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
  - Log security-related events and check logs
- ◆ Intrusion detection
  - Anomaly detection
    - Learn normal activity, Report abnormal actions
  - Attack detection
    - Recognize patterns associated with known attacks

52

## Interesting risk: data lifetime

- ◆ Recent work
  - Shredding Your Garbage: Reducing Data Lifetime Through Secure Deallocation  
by Jim Chow, Ben Pfaff, Tal Garfinkel, Mendel Rosenblum
- ◆ Example
  - User types password into web form
  - Web server reads password
  - Where does this go in memory?
    - Many copies, on stack and heap
    - Optimizing compilers may remove "dead" assignment/memcopy
    - Presents interesting security risk

53

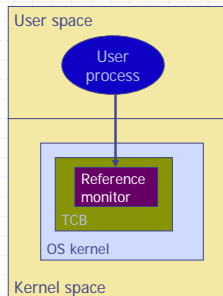
## Sample Features of Trusted OS

- ◆ 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
  - Log security-related events and check logs
- ◆ Intrusion detection (cover in another lecture)
  - Anomaly detection
    - Learn normal activity, Report abnormal actions
  - Attack detection
    - Recognize patterns associated with known attacks

54

## 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



55

## 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 ...

56

## 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

57

## 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)
- ◆ Replaces TCSEC, endorsed by 14 countries
  - CC adopted 1998
  - Last TCSEC evaluation completed 2000

<http://www.commoncriteria.org/>

58

## 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

59

## 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

60

## 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

61

## 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

62



63

## 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
- Security depends on installation as well as system

64

## Secure attention sequence (SAS)

### ◆ CTRL+ALT+DEL

- "... can be read only by Windows, ensuring that the information in the ensuing logon dialog box can be read only by Windows. This can prevent rogue programs from gaining access to the computer."

### ◆ How does this work?

- Winlogon service responds to SAS
- DLL called GINA (for Graphical Identification 'n' Authentication) implemented in msgina.dll gathers and marshals information provided by the user and sends it to the Local Security Authority (LSA) for verification
- The SAS provides a level of protection against Trojan horse logon prompts, but not against driver level attacks.

65

## Summary

### ◆ Access Control Concepts

- Matrix, ACL, Capabilities
- Multi-level security (MLS)

### ◆ OS Mechanisms

- Multics
  - Ring structure
- Amoeba
  - Distributed, capabilities
- Unix
  - File system, Setuid
- Windows
  - File system, Tokens, EFS
- SE Linux
  - Role-based, Domain type enforcement

### ◆ Assurance, Limitations

- Secure OS
  - Methods for resisting stronger attacks
- Assurance
  - Orange Book, TCSEC
  - Common Criteria
  - Windows 2000 certification
- Some Limitations
  - Information flow
  - Covert channels

66