CIS 551 / TCOM 401

Computer and Network Security

Spring 2009
Lecture 11

Announcements

- Plan for Today:
  - Access Control
  - Discretionary vs. Mandatory access control
  - Software validation

- Project 2 reminder
  - Due: Friday, March 6th (right before Spring Break)

Access Control Matrices

A[s][o]	Obj_1	Obj_2	…	Obj_N
Subj_1	{r,w,x}	{r,w}	…	{}
Subj_2	{w,x}	{}	…	
…	…	…	…	
Subj_M	{x}	{r,w,x}	…	{r,w,x}

Each entry contains a set of rights.

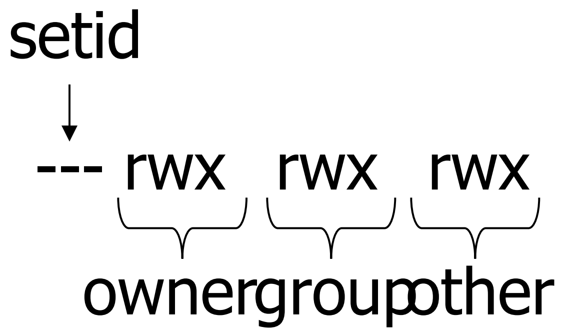
Access Control Lists

A[s][o]	Obj_1	Obj_2	…	Obj_N
$Subj_1$	{r,w,x}	{r,w}	…	{}
$Subj_2$	{w,x}	{}	…	{r}
…	…	…	…	…
$Subj_M$	{x}	{r,w,x}	…	{r,w,x}

For each object, store a list of (Subject x Rights) pairs.

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



Setid bits on executable Unix file

- Three setid bits
  - 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
  - Setuid – set EUID of process to ID of file owner
      - passwd owned by root and setuid is true
      - Jeff executes passwd: “passwd runs as root”
  - Setgid – set EGID of process to GID of file

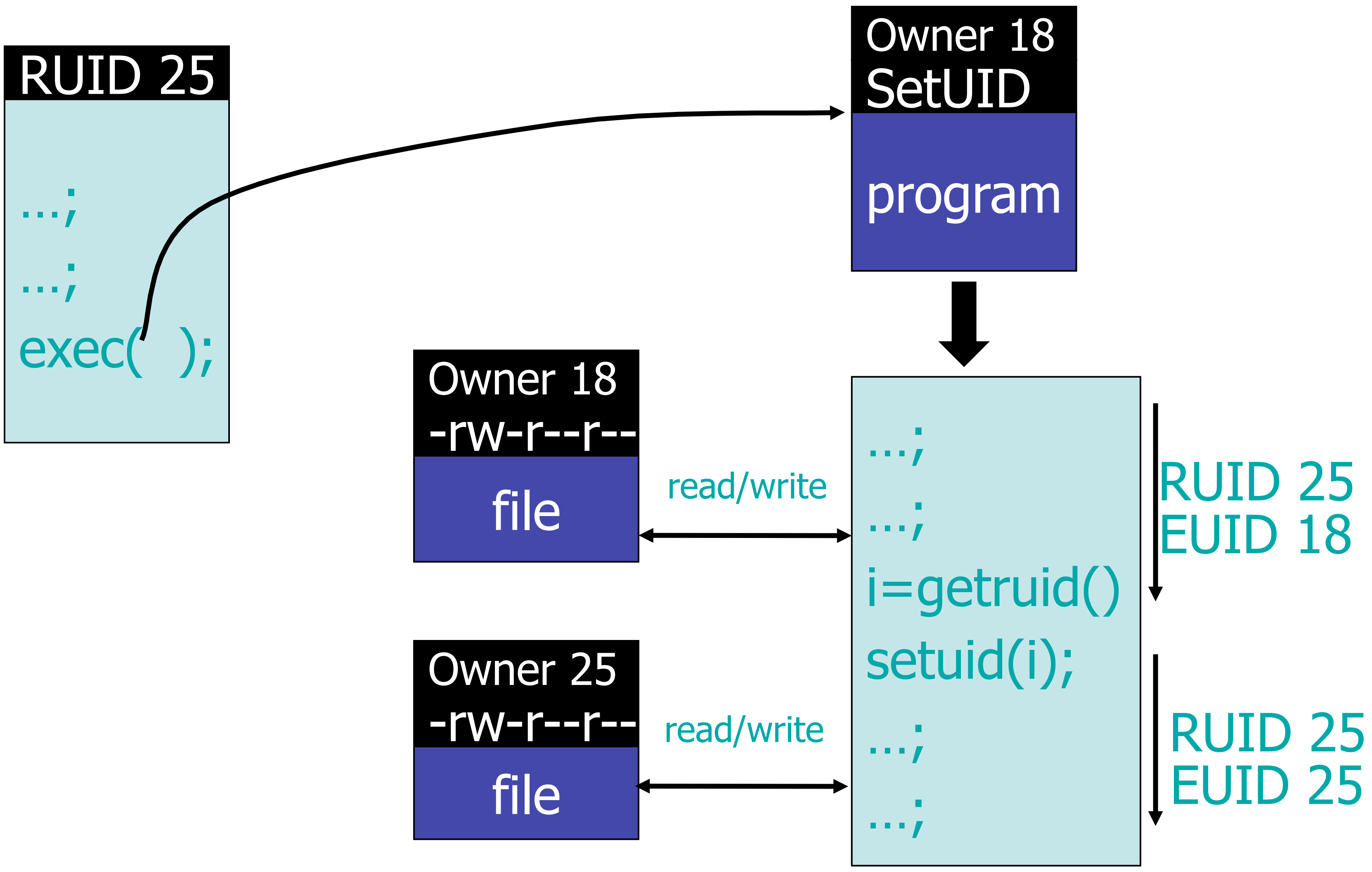
Effective User ID (EUID)

- Each process has three user IDs (more in 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 program file, or system 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 when executing a file with setuid bit on.
- 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, setruid

Example


RUID 25
…;
…;
exec( );
Owner 18
SetUID
program
Owner 18
-rw-r--r--
file
read/write
Owner 25
-rw-r--r--
file
read/write
…;
…;
i=getruid()
setuid(i);
…;
…;
RUID 25
EUID 18
RUID 25
EUID 25

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

Capabilities Lists

A[s][o]	Obj_1	Obj_2	…	Obj_N
$Subj_1$	{r,w,x}	{r,w}	…	{}
$Subj_2$	{w,x}	{}	…	{r}
…	…	…	…	…
$Subj_M$	{x}	{r,w,x}	…	{r,w,x}

For each subject, store a list of (Object x Rights) pairs.

Capabilities

- A capability is a (Object, Rights) pair
  - Used like a movie ticket e.g.: ("Slumdog Millionaire"@7:00pm, {admit one})
- Should be unforgeable
  - Otherwise, subjects could get illegal access
- Authentication takes place when the capabilities are granted (not needed at use)
- Harder to do revocation (must find all tickets)
- Easy to audit a subject, hard to audit an object

Implementing Capabilities

- Must be able to name objects
- Unique identifiers
  - Must keep map of UIDs to objects
  - Must protect integrity of the map
  - Extra level of indirection to use the object
  - Generating UIDs can be difficult
- Pointers
  - Name changes when the object moves
  - Remote pointers in distributed setting
  - Aliasing possible

Unforgeability of Capabilities

- Special hardware: tagged words in memory
  - Can’t copy/modify tagged words
- Store the capabilities in protected address space
- Could use static scoping mechanism of safe programming languages.
  - Java’s “private” fields
- Could use cryptographic techniques
  - OS kernel could sign (Object, Rights) pairs using a private key
  - Any process can verify the capability
  - Example: Kerberos

Access control in Windows (NTFS)

- Some basic functionality similar to Unix
  - Specify access for groups and users
    - Read, modify, change owner, delete
  - ACLs used for fine grained control
- Some additional concepts
  - Security Tokens -- these are a form of capability
    - Allows clients to temporarily upgrade their privileges
    - Can also be used for delegation of privileges from one process to another
    - Replaces Unix's setuid model
  - Security attributes
- Generally
  - More flexibility than Unix
    - Can define new permissions
    - Can give some but not all administrator privileges

Access Control

- *Discretionary*: The individual user may, at his own discretion, determine who is authorized to access the objects he creates.

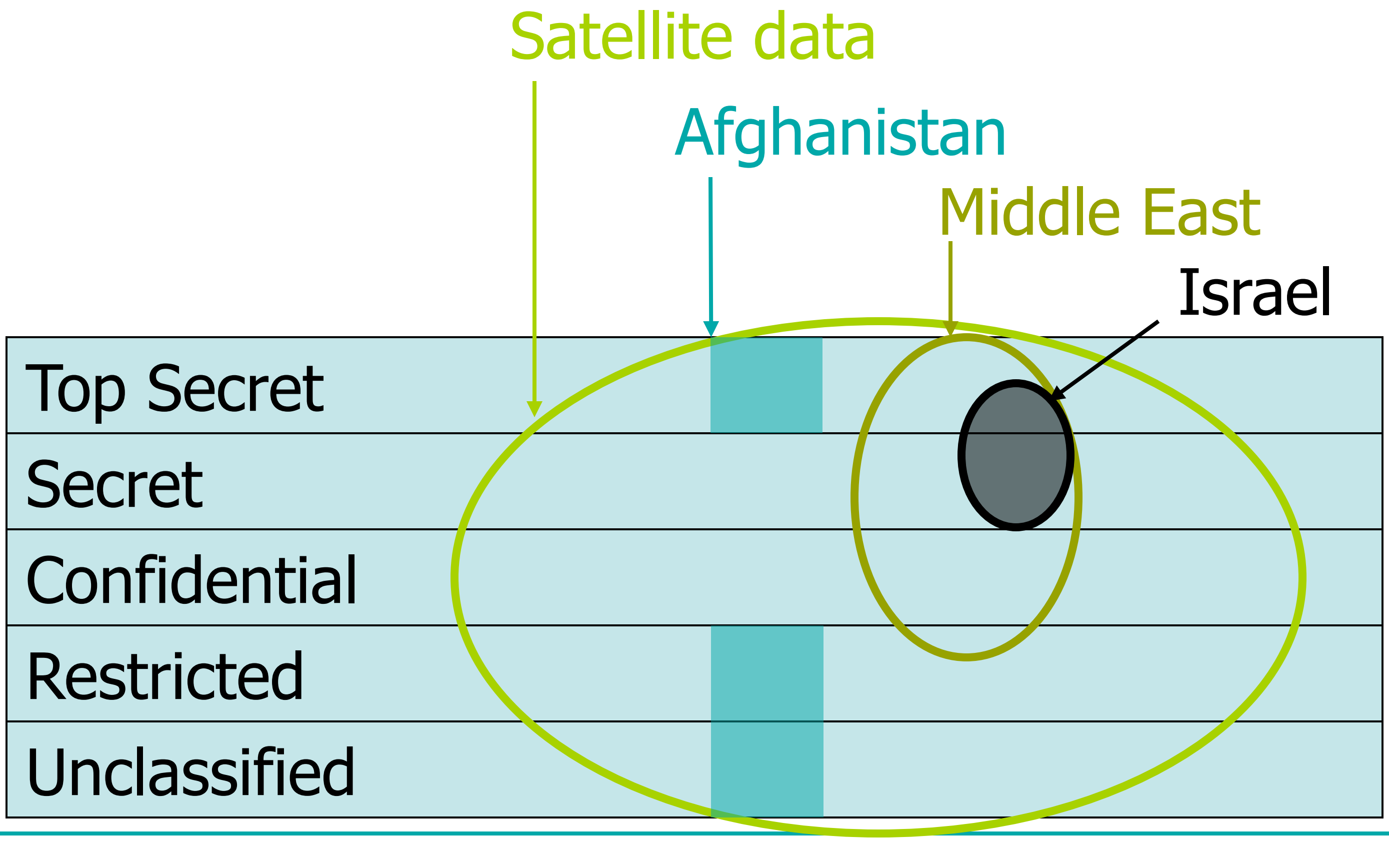
- *Mandatory*: The creator of an object does not necessarily have the ability to determine who has authorized access to it.
  - Typically policy is governed by some central authority
  - The policy on an object in the system depends on what object/ information was used to create the object.

Multilevel Security

- Multiple levels of confidentiality or integrity ratings
- 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
- Trivial example: Public ≤ Secret
- *Information flow*
    - Regulate how information is used throughout entire system
    - A document generated from both Public and Secret information must be rated Secret.
    - Intuition: "Secret" information should not flow to "Public" locations.

Military security policy

- Sensitivity levels
- Compartments



Military security policy

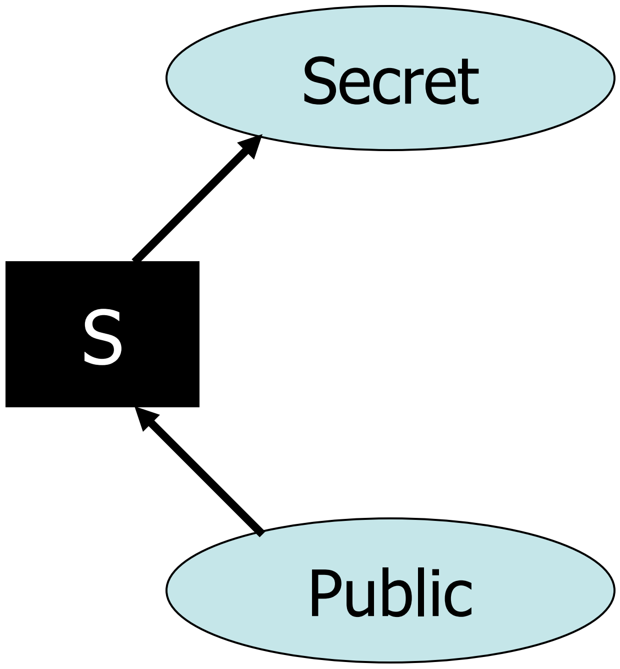
- Classification of personnel and data
  - Class D = $\langle$rank, compartment$\rangle$
- Dominance relation
  - $D_1 \leq D_2$ iff $\text{rank}_1 \leq \text{rank}_2$
    and $\text{compartment}_1 \subseteq \text{compartment}_2$
  - Example: $\langle$Restricted, Israel$\rangle$ $\leq$ $\langle$Secret, Middle East$\rangle$
- Applies to
  - Subjects – users or processes: C(S) = "clearance of S"
  - Objects – documents or resources: C(O) = "classification of O"

Bell-LaPadula Confidentiality Model

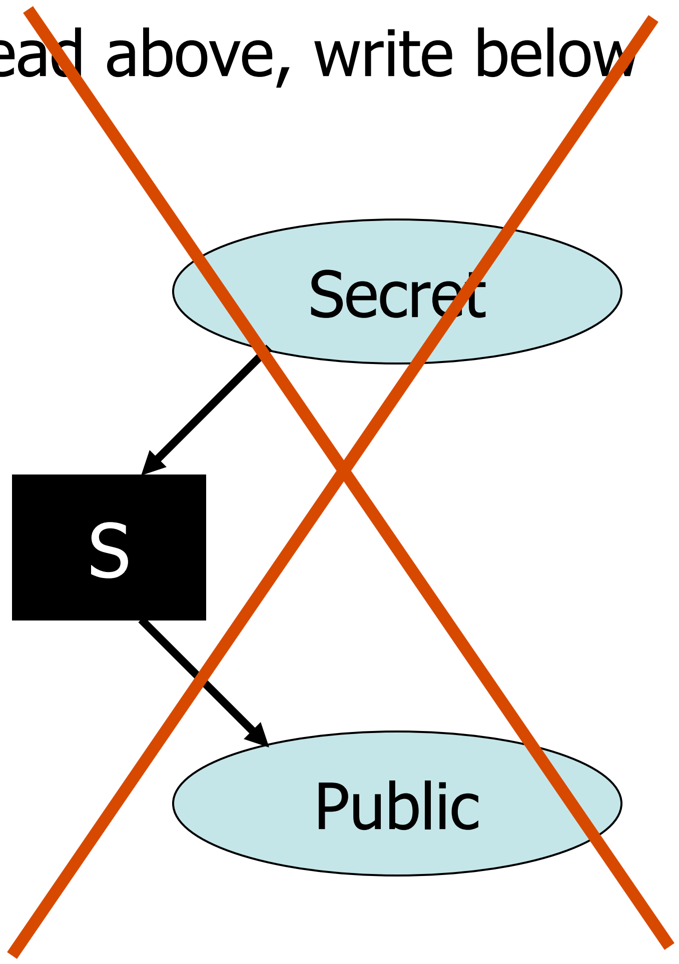
- “No read up, no write down.”
  - Subjects are assigned clearance levels drawn from the lattice of security labels.
    - C(S) = "clearance of the subject S"
  - A principal may read objects with lower (or equal) security label.
    - Read: $C(O) \leq C(S)$
  - A principal may write objects with higher (or equal) security label.
    - Write: $C(S) \leq C(O)$
- Example: A user with Secret clearance can:
  - Read objects with label Public and Secret
  - Write/create objects with label Secret

Picture: Confidentiality

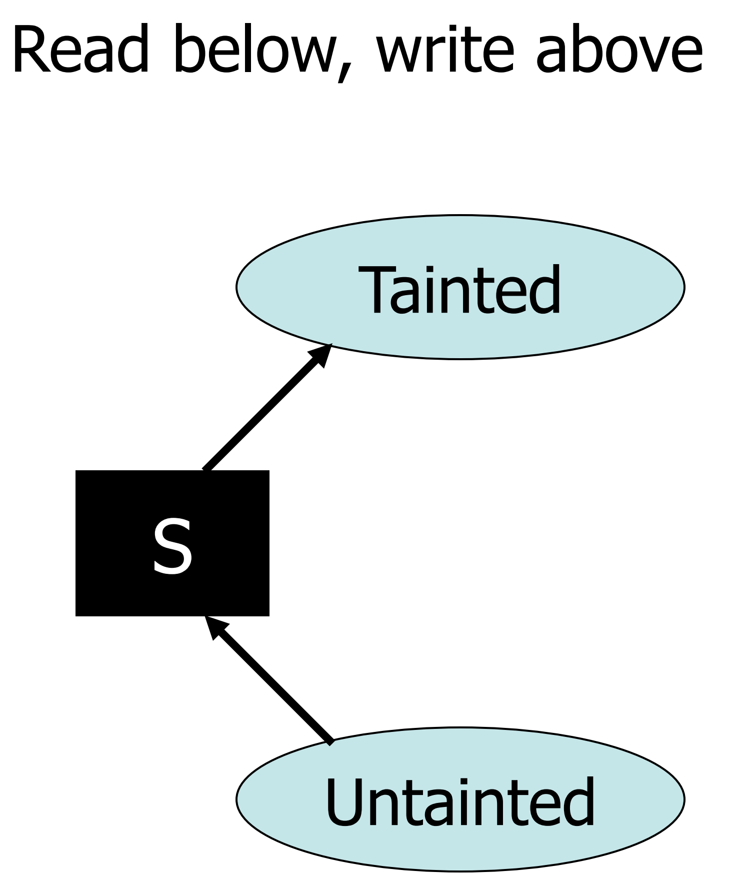
Read below, write above

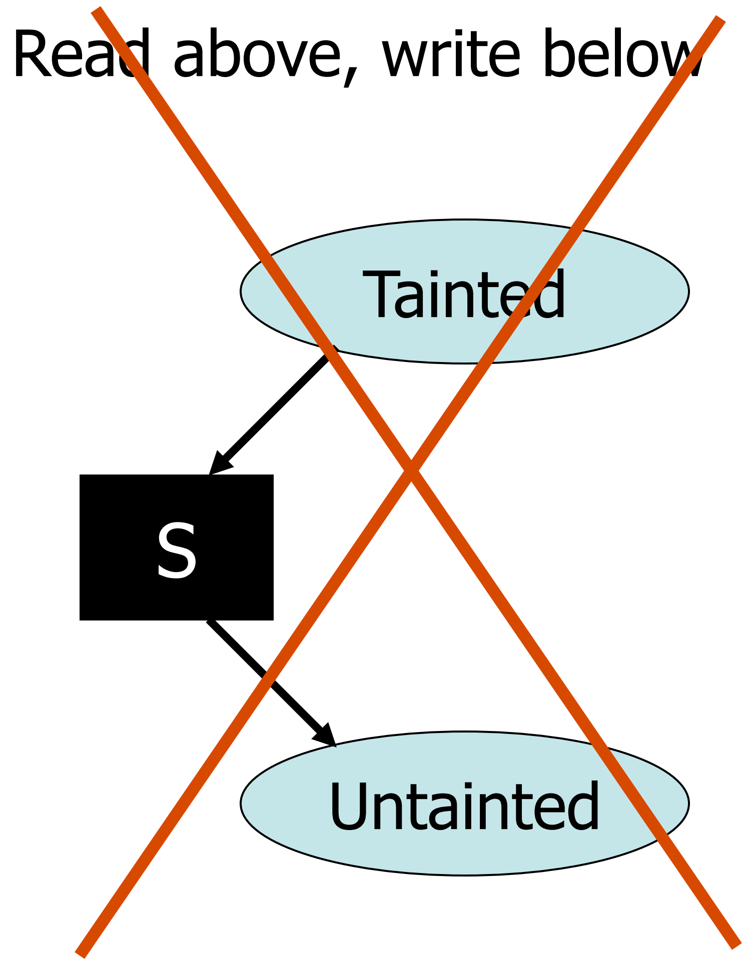


Read above, write below



Picture: Integrity





Multilevel Security Policies

- In general, security levels form a "join semi-lattice"
  - There is an ordering $\leq$ on security levels
  - For any pair of labels L1 and L2 there is an "join" operation:

    $L1 \oplus L2$ is a label in the lattice such that:
    (1) $L1 \leq L1 \oplus L2$ and $L2 \leq L1 \oplus L2$ "upper bound"
    (2) If $L1 \leq L3$ and $L2 \leq L3$ then $L1 \oplus L2 \leq L3$ "least bound"

- For example: Public $\oplus$ Secret = Secret
- Labeling rules:
  - Classification is a function $C : \text{Object} \rightarrow \text{Lattice}$
  - If some object O is "created from" objects $O_1,\ldots,O_n$ then $C(O) = C(O_1) \oplus \ldots \oplus C(O_n)$

Implementing Multilevel Security

- Dynamic:
  - Tag all values in memory with their security level
  - Operations propagate security levels
  - Must be sure that tags can’t be modified
  - Expensive, and approximate

- Classic result: Information-flow policies cannot be enforced purely by a reference monitor!
  - Problem arises from implicit flows

- Static:
  - Program analysis
  - May be more precise
  - May have less overhead

Information Flows through Software

Explicit Flows:

```
int{Secret} X = f();
int{Public} Y = 0;

Y = X;
```

Implicit Flows:

```
int{Secret} X = f();
int{Public} Y = 0;
int{Public} Z = 0;
int{Public} W = 0;

if (X > 0) then {
  Y = 1;
} else {
  Z = 1;
}
W = 3;
```

Perl's Solution (for Integrity)

- The problem: need to track the source of data
- Examples: Format string, SQL injection, etc.

```
$arg = shift;
system ("echo $arg");
```

- Give this program the argument `"; rm *"`
- Perl offers a *taint checking* mode
  - Tracks the source of data (trusted vs. tainted)
  - Ensure that tainted data is not used in system calls
  - Tainted data can be converted to trusted data by pattern matching
  - Doesn't check implicit flows

SELinux

http://www.nsa.gov/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
- Type enforcement
  - Each process has an associated domain
  - Each object has an associated type (label)
  - 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

Two Other MAC Policies

- "Chinese Wall" policy: [Brewer & Nash '89]
  - Object labels are classified into "conflict classes"
  - If subject accesses one object with label L1 in a conflict class, all access to objects labeled with other labels in the conflict class are denied.
  - Policy changes dynamically
- "Separation of Duties":
  - Division of responsibilities among subjects
  - Example: Bank auditor cannot issue checks.

Question:

- Suppose you have gone through the cost/benefit and risk analysis to determine the securty requirements for a computer system.
- How do you know whether a system meets its security requirements?

- Class answers:

Assurance methods

- Testing
  - Regression testing, automation tools, etc.
  - Can demonstrate existence of flaw, not absence
- Validation
  - Requirements checking
  - Design and code reviews
    - Sit around table, drink lots of coffee, …
  - Module and system testing
- Formal verification
  - Develop a rigorous (mathematical) specification of the system
  - Prove (using tools or by hand) that the implementation meets the specification
  - Time-consuming, painstaking process
  - Has been done for some systems. (See www.praxis-his.com)

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.fas.org/irp/nsa/rainbow.htm

Orange Book Requirements (TCSEC)

- TCSEC = Trusted Computer System Evaluation Criteria

- 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

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

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.niap-ccevs.org/cc-scheme/

http://www.commoncriteriaportal.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, including high-level design
- Independent testing, review of developer testing

EAL 3: Methodically Tested and Checked

- Development environment controls; configuration 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

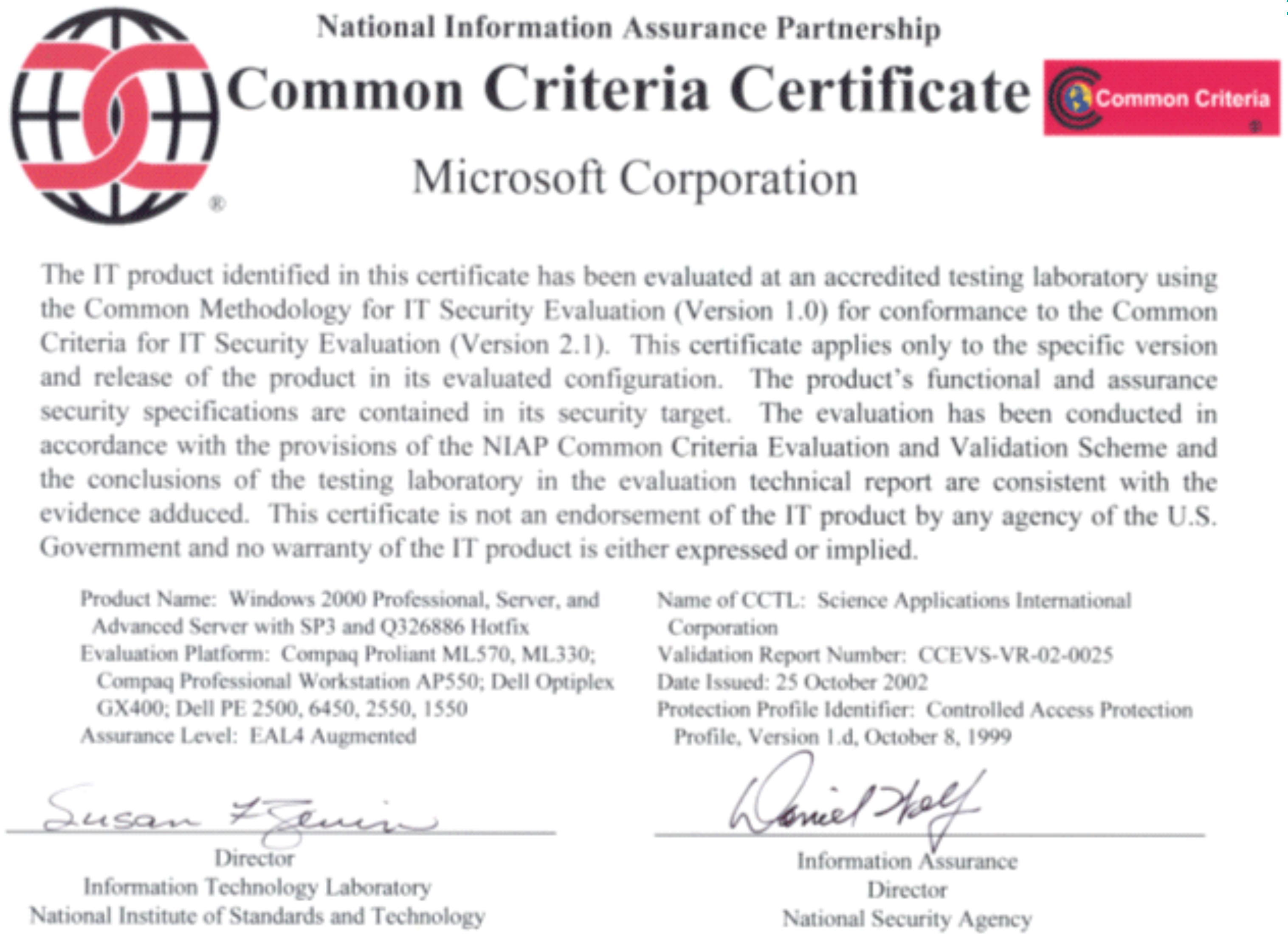
EAL 7: Formally Verified Design and Tested

- 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



National Information Assurance Partnership

Common Criteria Certificate

Microsoft Corporation

The IT product identified in this certificate has been evaluated at an accredited testing laboratory using the Common Methodology for IT Security Evaluation (Version 1.0) for conformance to the Common Criteria for IT Security Evaluation (Version 2.1). This certificate applies only to the specific version and release of the product in its evaluated configuration. The product's functional and assurance security specifications are contained in its security target. The evaluation has been conducted in accordance with the provisions of the NIAP Common Criteria Evaluation and Validation Scheme and the conclusions of the testing laboratory in the evaluation technical report are consistent with the evidence adduced. This certificate is not an endorsement of the IT product by any agency of the U.S. Government and no warranty of the IT product is either expressed or implied.

Product Name: Windows 2000 Professional, Server, and Advanced Server with SP3 and Q326886 Hotfix
Evaluation Platform: Compaq Proliant ML570, ML330; Compaq Professional Workstation AP550; Dell Optiplex GX400; Dell PE 2500, 6450, 2550, 1550
Assurance Level: EAL4 Augmented

Name of CCTL: Science Applications International Corporation
Validation Report Number: CCEVS-VR-02-0025
Date Issued: 25 October 2002
Protection Profile Identifier: Controlled Access Protection Profile, Version 1.d, October 8, 1999

[illegible]
Director
Information Technology Laboratory
National Institute of Standards and Technology

[illegible]
Information Assurance
Director
National Security Agency