CIS 551 / TCOM 401

Computer and Network Security

Spring 2009
Lecture 12

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

- *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.

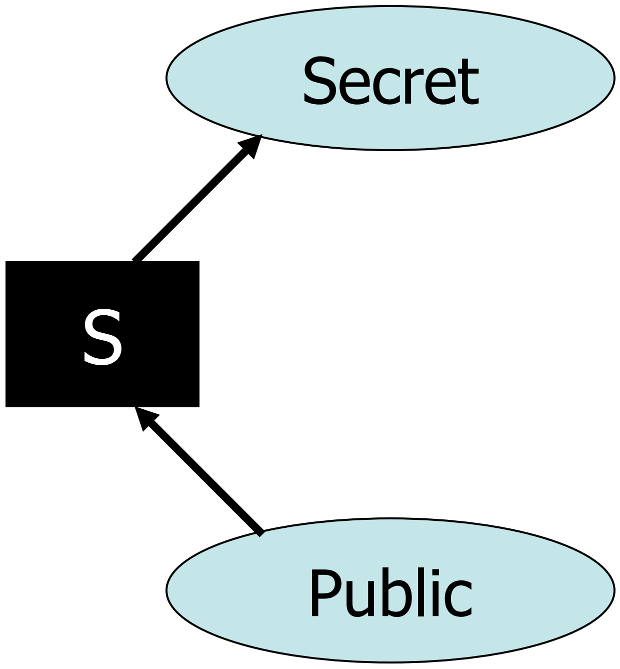
Bell-LaPadula Confidentiality Model

- “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: $C(S) \leq C(O)$

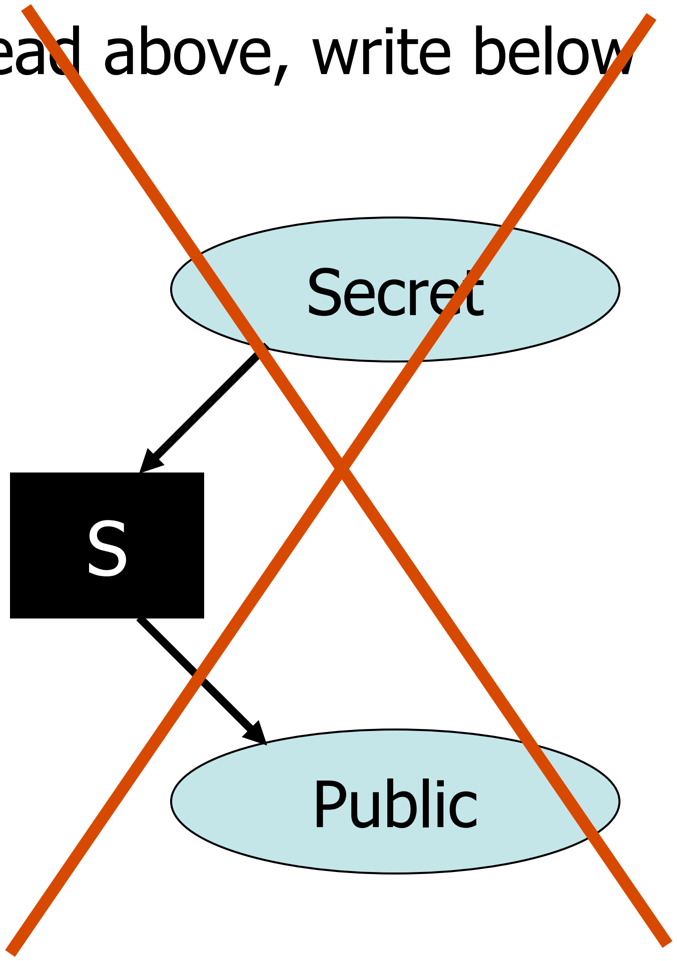
- Example: A user with Secret clearance can:
  - Read objects with label Public and Secret
  - Write/create objects with label Secret

Picture: Confidentiality

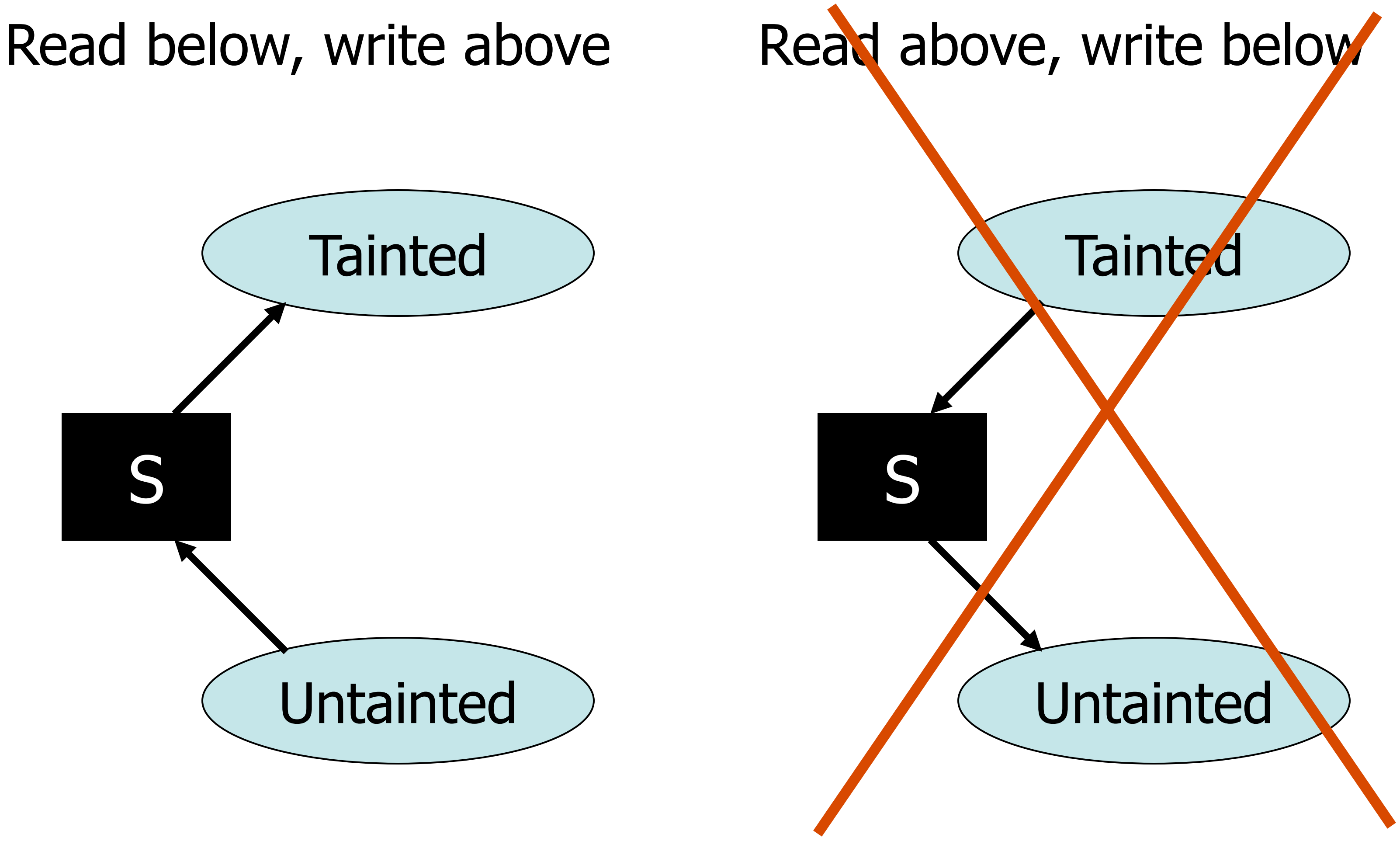
Read below, write above



~~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

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

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;
```

Jif: Java+Information Flow

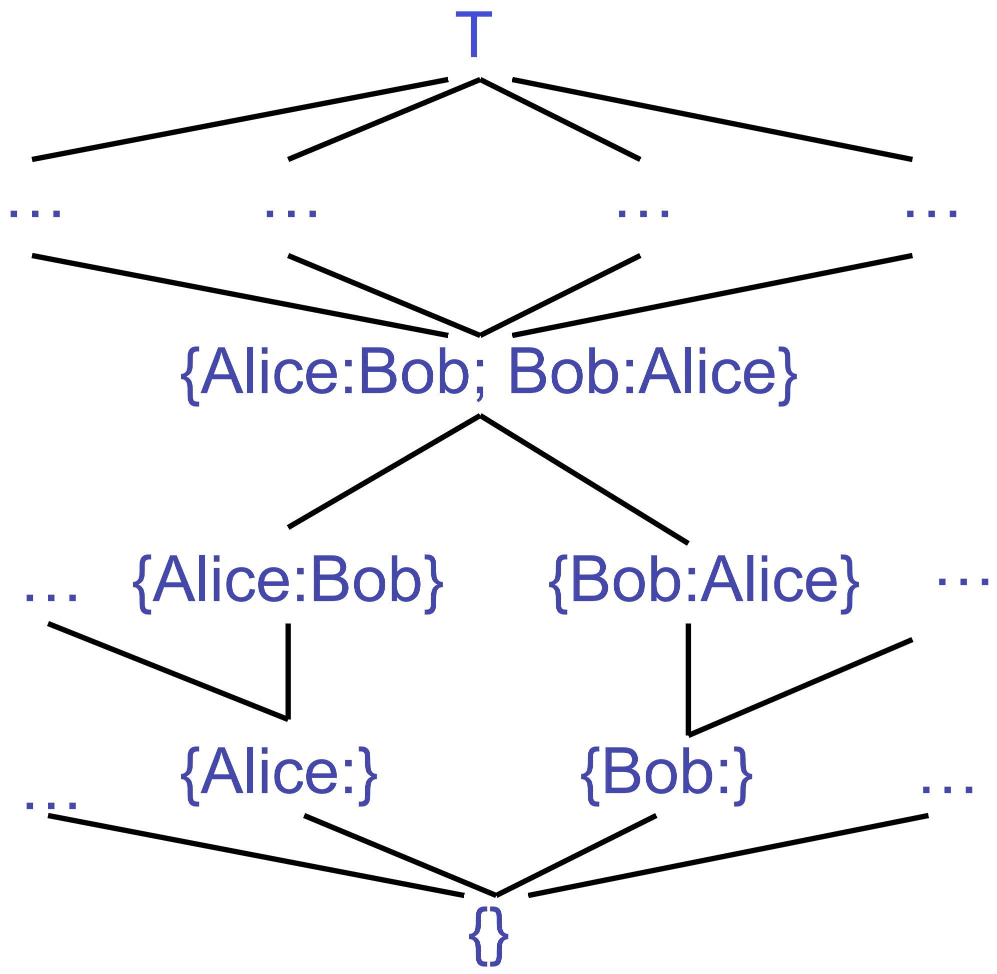
[Myers, Chong, Nystrom, Zdancewic, Zheng] http://www.cs.cornell.edu/jif/

- Policy Language that extends Java’s type system
  - Confidentiality & Integrity constraints
  - Principal hierarchy (delegation)
  - Robust Declassification
- Jif is intended to enforce *noninterference*
  - Intuitively, requires that high security data not affect any behavior of the program that is visible to low-clearance users.

Decentralized Label Model

[Myers & Liskov '97, '00]

- Principals: users, groups, etc.
  - Express constraints on data usage
  - Distinct from hosts
  - Alice, Bob, etc. are principals
- Labels:
  - {} bottom label (fewest constraints)
  - {Alice: Bob} readably by Alice and Bob
  - {Alice: Bob, Charles; Bob: Charles} readable by Charles
  - {o$_1$: r$_{11}$,…,r$_{1k}$; …; o$_n$: r$_{n1}$,..r$_{nm}$}



Jif Policy Annotations

- Augment Java types with labels:

```
int{Alice:} a;
int{Bob:}  b;
int{Alice:Bob:} c;
```

- Give appropriate types to I/O routines:

```
void Network.send(int{} x)
```

- Type-checking detects illegal flows:

```
a = b;
Network.send(a);
c = a + b;
```

Security Policies in Jif

- Confidentiality labels:
  ```
  int{Alice:} a;          "a is Alice's private int"
  ```
- Integrity labels:
  ```
  int{?Alice} a;          "Alice must trust a"
  ```
- Compound labels:
  ```
  int{Alice: ?Alice} a;          (Both constraints)
  ```

```
                              // Insecure        // Secure
int{Alice:} a1, a2;           a1 = b;            a1 = a2;
int{Bob:} b;                  b = a1;
```

Richer Security Policies

- More complex policies:

  "Alice will release her data to Bob, but only after he has paid her $10."

- Declassification
  - Escape from strict noninterference
  - Like “cast” in C, it’s dangerous
  - Bound its effects
- Jif uses the notion of *authority*
  - Program runs on behalf of set of principals
  - Classes and methods can declare authority requirements

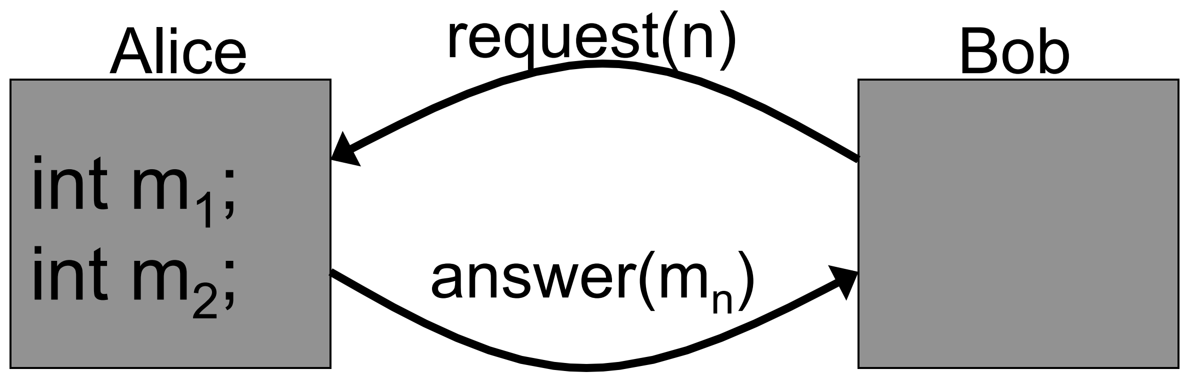
Declassification

```
int{Alice:} a;
int Paid;
...   // compute Paid
if (Paid==10) {
      int{Bob:} b = declassify(a to Bob);
      ...
}
```

"type-cast"
int{Alice:} to
int{Bob:}

Example: Oblivious Transfer

[Rabin '81]



- Alice's Policy:

  "Bob gets to choose exactly one of m_1 and m_2."
- Bob's Policy:

  "Alice doesn't get to know which item I request."
- Classic Result: "Impossible to solve using 2 principals, with perfect security."

[Damgård, Kilian, Slavail '99]

Oblivious Transfer (Java)

```
int               m1, m2;   // Alice's data
boolean           accessed;
int               n, ans;   // Bob's data

n = choose();               // Bob's choice

if (!accessed) {            // Transfer
   accessed = true;
   if (n == 1)
        ans = m1;
   else ans = m2;
}
```

Adding Confidentiality Labels

```
int{Alice:}          m1, m2;   // Alice's data
boolean            accessed;
int{Bob:}             n, ans;   // Bob's data
  ____________________
n = choose();                   // Bob's choice

if (!accessed) {                // Transfer
   accessed = true;
   if (n == 1)
        ans = m1;
   else ans = m2;
}
```

Verification Fails

Using Declassification

```
int{Alice}          m1, m2;    // Alice's data
boolean             accessed;
int{Bob}            n, ans;    // Bob's data

n = choose();                          s choice

if (!accessed) {                   // Transfer
   accessed = true;
   if (n == 1)
        ans = declassify(m1 to Bob);
   else ans = declassify(m2 to Bob);
}
```

Verification Fails

Integrity Constraints

```
int{Alice}         m1, m2;    // Alice's data
boolean{Alice?}    accessed;
int{Bob}           n, ans;    // Bob's data
___________________

n = choose();                 // Bob's choice

if (!accessed) {
   accessed = true;
   if (n == 1)
        ans = declassify(m1 to Bob);
   else ans = declassify(m2 to Bob);
}
```

Verification Fails

Using Endorsement

```
int{Alice}          m1, m2;    // Alice's data
boolean{Alice?}     accessed;
int{Bob}            n, ans;    // Bob's data

n = choose();                  // Bob's choice

if (!accessed) {               // Transfer
   accessed = true;
   if (endorse(n by Alice) == 1)
        ans = declassify(m1 to Bob);
   else ans = declassify(m2 to Bob);
}
```

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.

Covert Channels & Information Hiding

- A covert channel is a means by which two components of a system that are not permitted to communicate do so anyway by affecting a shared resource.

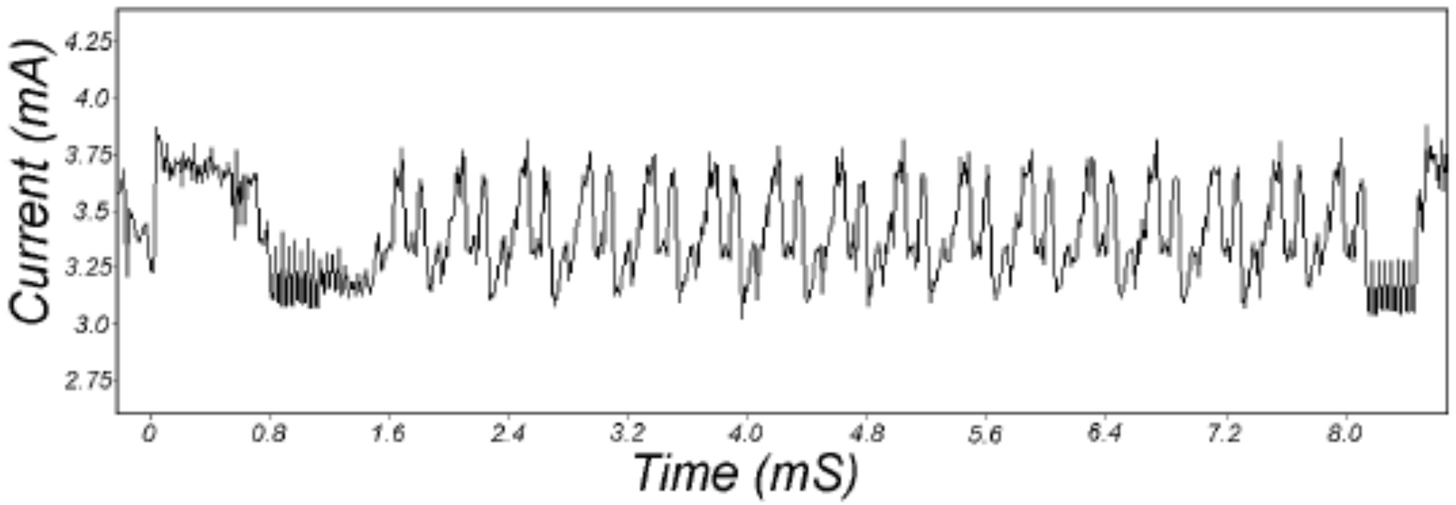
- Information hiding: Two components of the system that are permitted to communicate about one set of things, exchange information about disallowed topics by encoding contraband information in the legitimate traffic.

- Not that hard to leak a small amount of data
  - A 64 bit encryption key is not that hard to transmit
  - Even possible to encode relatively large amounts of data!

- Example channels / information hiding strategies
  - Program behavior
  - Adjust the formatting of output:
    use the “\t” character for “1” and 8 spaces for “0”
  - Vary timing behavior based on key
  - Use "low order" bits to send signals
  - Power consumption
  - Grabbing/releasing a lock on a shared resource

Differential Power Analysis

- Read the value of a DES password off of a smartcard by watching power consumption!



- This figure shows simple power analysis of DES encryption. The 16 rounds are clearly visible.

TEMPEST Security

- Transient Electromagnetic Pulse Emanation Standard
  - (Or?) Temporary Emanation and Spurious Transmission
  - Emission security (Van Eck phreaking)
  - computer monitors and other devices give off electromagnetic radiation
  - With the right antenna and receiver, these emanations can be intercepted from a remote location, and then be redisplayed (in the case of a monitor screen) or recorded and replayed (such as with a printer or keyboard).
- Policy is set in National Communications Security Committee Directive 4
- Guidelines for preventing EM reception
  - Shield the device (expensive)
  - Shield a location (inconvenient?)

Defenses for Covert Channels

- Well specified security policies at the human level
- Auditing mechanisms at the human level
  - Justify prosecution if the attacker is caught
- Code review
  - This is a form of audit
- Automated program analysis
  - Type systems that let programmers specify confidentiality labels
  - Transform programs so that both branches of a conditional statement take the same amount of time
  - Disallow branches on "secret" information
- Automated system analysis
  - Monitor http traffic to look for unusual behavior

Specific Countermeasures

- Against timing attacks:
  - Make all operations run in same amount of time
    - Hard to implement!
    - Can’t design platform-independent algorithms
    - All operations take as long as slowest one
  - Add random delays
    - Can take more samples to remove randomness
- Against power analysis attacks:
  - Make all operations take the same amount of power
    - Again, hard to implement
  - Add randomness

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

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

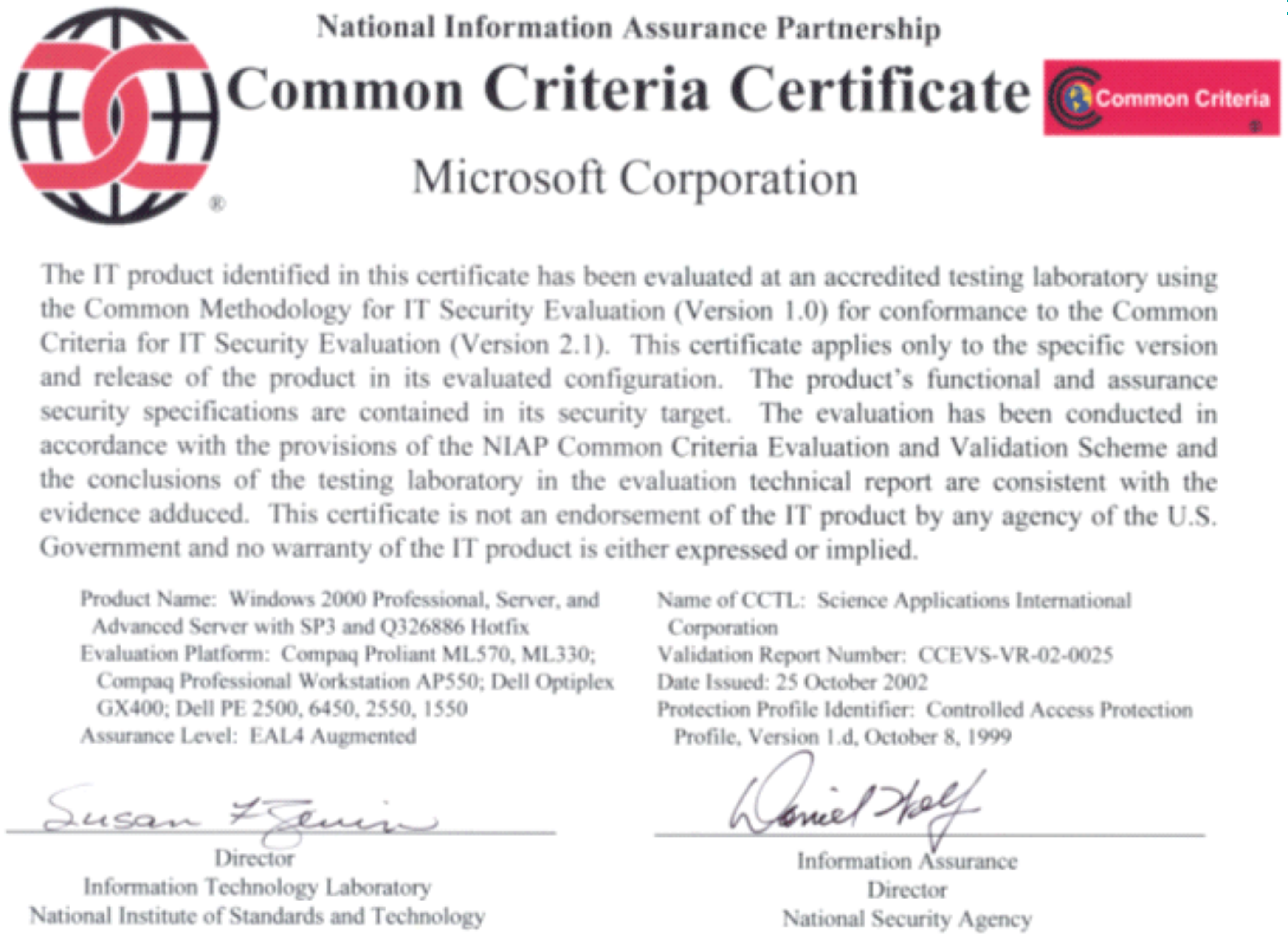
EAL 7: Formally Verified Design and Tested

- 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