

# IT 4823

## Information Security Administration

### Trusted Computing



Notice: This session is  
being recorded.

Some lecture slides prepared by Dr Lawrie Brown for “*Computer Security: Principles and Practice*”, 1/e, by William Stallings and Lawrie Brown,



Copyright © 2016 by Bob  
Brown



# Trusted Computing

## Components:

- Formal models for computer security
  - Access Control
  - Integrity
- Multilevel security
- Trusted systems
- Security evaluation and the Common Criteria

# Formal Models for Computer Security

- Two fundamental computer security facts:
  - All complex software systems contain errors
  - It is extraordinarily difficult to build computer hardware/software not vulnerable to attack
- Hence, the desire to prove design and implementation satisfy security requirements
- That led to development of formal security models; initially funded by US DoD
- The Bell-LaPadula (BLP) model was very influential

# Bell-LaPadula (BLP) Model

- Developed in 1970s as a formal access control model
- Subjects and objects have a **security class**
  - top secret > secret > confidential > unclassified
  - subject has a **security clearance** level
  - object has a **security classification** level
  - the class controls how a subject may access an object

# Simple Security Condition (Prelim)

- Information flows up, not down.
- “No read up”
- Subject  $S$  has security clearance  $C_s$
- Object  $O$  has security classification  $C_o$
- Subject  $S$  can read object  $O$  iff  $C_s \geq C_o$
- Example: Adam is cleared for “Secret”
  - Adam can read a document classified secret
  - Adam can read a document classified confidential
  - Adam *cannot* read a document classified top secret

## \*-Property (Prelim)

- Information flows up, not down
- “No write down”
- Subject  $S$  can write object  $O$  iff  $C_s \leq C_o$
- Example: Adam is cleared for “Secret”
  - Adam can write a document classified secret
  - Adam can write a document classified top secret Adam *cannot* write a document classified confidential
- Goal: to prevent leakage of information to lower levels.

# Basic Security Theorem, Step 1

If a system is initially in a secure state, and every transition of the system satisfies both the simple security condition, preliminary, and the \*-property, preliminary, then every state of the system is secure

# Adding Category Labels

- “Need to know”
- Add category labels to security clearances and classifications: (*classification, {categories}*)
- Example: An object is classified (*secret, {nuclear, Europe}*)
- Bill has clearance (*top secret, {nuclear, Asia}*)  
Bill cannot read the document (needs “Europe”)
- Charlie has clearance (*secret, {nuclear, Asia, Europe}*)  
Charlie **can** read the document



# The “Dominates” Relation

*S dominates O* iff

$Classification_s \geq Classification_o$  and

$Labels_s \supseteq Labels_o$

# Levels and Lattices

- $(C, L) \text{ dom } (C', L')$  iff  $C' \leq C$  and  $L' \subseteq L$
- Examples
  - $(\text{Top Secret}, \{\text{NUC}, \text{ASIA}\}) \text{ dom } (\text{Secret}, \{\text{NUC}\})$
  - $(\text{Secret}, \{\text{NUC}, \text{EUR}\}) \text{ dom } (\text{Confidential}, \{\text{NUC}, \text{EUR}\})$
  - $(\text{Top Secret}, \{\text{NUC}\}) \neg \text{dom } (\text{Confidential}, \{\text{EUR}\})$
- Let  $C$  be set of classifications,  $L$  set of labels.  
The set of security levels  $K = C \times L$ ,  $\text{dom}$  form lattice
  - $\text{lub}(K) = (\max(C), L)$
  - $\text{glb}(K) = (\min(C), \emptyset)$

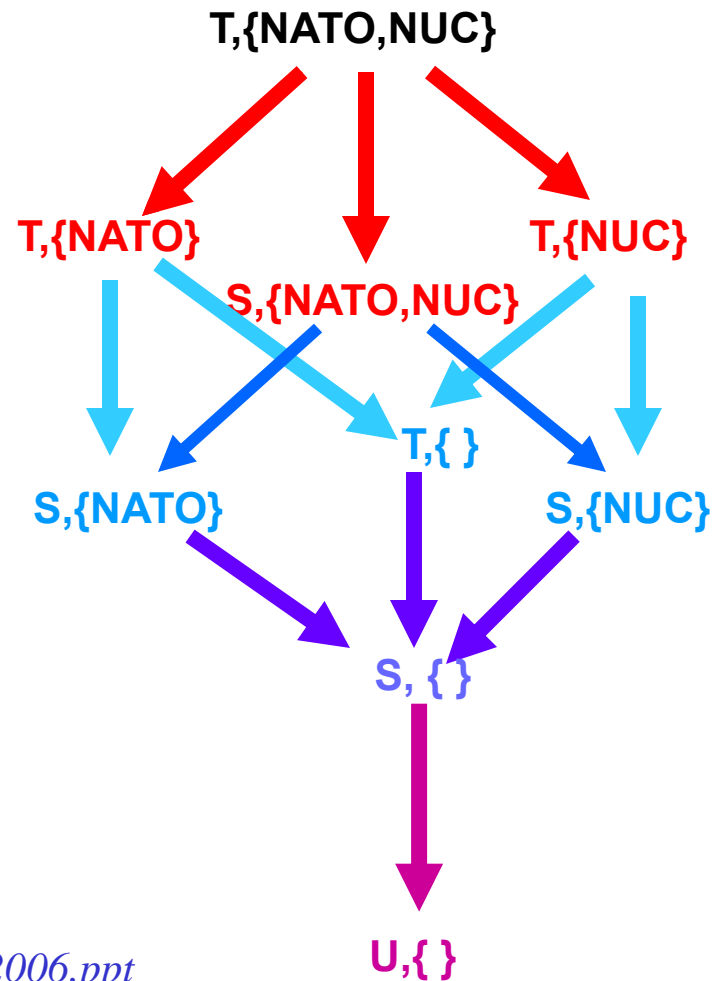
# Bell-LaPadula Lattice

Classifications: T, S, U

Labels: NATO, NUC

Least upper bound:  
 $T, \{NATO, NUC\}$

Greatest Lower Bound:  
 $U, \{\}$



[www.cs.virginia.edu/~jones/cs451/slides/info%20flow%2006.ppt](http://www.cs.virginia.edu/~jones/cs451/slides/info%20flow%2006.ppt)

# Reading Information

- Information flows *up*, not *down*
  - “Reads up” disallowed, “reads down” allowed
- Simple Security Condition (Step 2)
  - $C$  is a classification, top secret, secret, etc.
  - $L$  is a set of “need to know” labels
  - Subject  $s$  can read object  $o$  iff  $(C,L)_s \text{ dom } (C,L)_o$
  - We can include discretionary access control by adding, “and  $s$  has permission to read  $o$ . ”
  - Sometimes called “no read up” rule

# Writing Information

- Information flows up, not down
  - “Writes up” allowed, “writes down” disallowed
- \*-Property (Step 2)
  - Subject  $s$  can write object  $o$  iff  $(C,L)_o \text{ dom } (C,L)_s$  and  $s$  has permission to write  $o$
  - Sometimes called “no write down” rule

## Basic Security Theorem, Step 2

- If a system is initially in a secure state, and every transition of the system satisfies the simple security condition, step 2, and the \*-property, step 2, then every state of the system is secure

# Problem

- Colonel has (Secret, {NUC, EUR}) clearance
- Major has (Secret, {EUR}) clearance
  - Major can talk to colonel (“write up” or “read down”)
  - Colonel cannot talk to major (“read up” or “write down”)
- Clearly absurd!

# Solution

- Define maximum, current levels for subjects
  - $maxlevel(s) \text{ dom } curlevel(s)$
- Example
  - Treat Major as an object (Colonel is “writing”)
  - Colonel has  $maxlevel$  (Secret, { NUC, EUR })
  - Colonel sets  $curlevel$  to (Secret, { EUR })
  - Now  $(C, L)_{\text{Major}} \text{ dom } curlevel_{\text{Colonel}}$  Colonel can write to Major without violating “no write down.”
  - Does  $(C, L)_s$  mean  $curlevel(s)$  or  $maxlevel(s)$ ?  
Formally, we need a more precise notation



# Integrity Policy Models

- Requirements:
  - Very different from confidentiality policies
  - We are describing whether (and how much) we can trust data. (Not who can see it.)
- Integrity: the state that exists when records agree with the source from which they were derived and have not been incorrectly altered or destroyed.
- Biba's model
- Clark-Wilson model

# Intuition for Integrity Levels

- The higher the level, the more confidence:
  - That a program will execute correctly
  - That data is accurate and/or reliable
- Note: there is a relationship between integrity and trustworthiness.
- Intuition: a less-trusted process (subject) should not modify a highly-trusted object.
- Important point: *integrity levels are **not** confidentiality levels*. Integrity is concerned with trustworthiness, not data flow.

# Biba Integrity Model

- Controls modification of data
- Cannot modify an object of higher integrity
- Cannot read object of lower integrity (no reading junk.) This is the “integrity confinement” property.
- Example: Integrity levels in MS Windows:
  - Browser is a low-integrity process
  - Cannot modify high-integrity files.
  - So, a vulnerability in the browser cannot be used to modify files belonging to the O.S.

# Clark-Wilson Integrity Model

- Possibly more suited to business than Biba.
- Integrity defined by a set of constraints
  - Data in a *consistent* or valid state when it satisfies constraints.
- Example: Bank
  - $D$  today's deposits,  $W$  withdrawals,  $YB$  yesterday's balance,  $TB$  today's balance
  - Integrity constraint:  $YB + D - W = TB$
- *Well-formed transactions* move system from one consistent state to another
- Issue: who examines, certifies transactions done correctly?

# Chinese Wall Model

## Problem:

- Tony advises Bank of America about investments
- He is asked to advise CitiBank about investments
- It is a conflict of interest to accept, because his advice for either bank would affect his advice to the other bank

# Organization

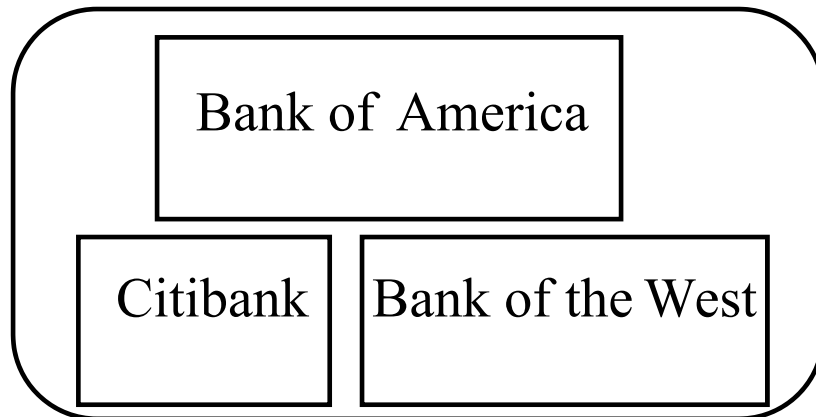
- Organize entities into “conflict of interest” classes
- Control subject accesses to each class
- Control writing to all classes to ensure information is not passed along in violation of rules
- Allow sanitized (public) data to be viewed by everyone

# Definitions

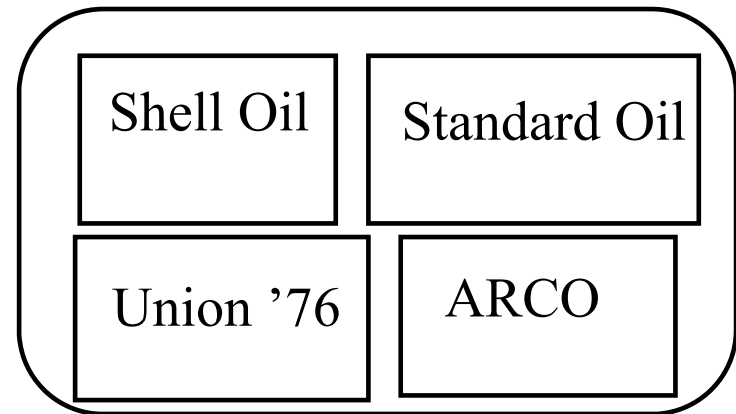
- *Objects*: items of information related to a company
- *Company dataset* (CD): contains objects related to a single company  
Written  $CD(O)$
- *Conflict of interest class* (COI): contains datasets of companies in competition  
Written  $COI(O)$
- Assumption: each object belongs to exactly one *COI* class

# Example

Bank COI Class



Gasoline Company COI Class





# Temporal Element

- If Anthony reads any CD in a COI, he can *never* read another CD in that COI
  - Possible that information learned earlier may allow him to make decisions later
  - Let  $PR(S)$  be set of objects that  $S$  has already read; initially the empty set.

# CW-Simple Security Condition

- $s$  can read  $o$  iff either condition holds:
  1. There is an  $o'$  such that  $s$  has accessed  $o'$  and  $CD(o') = CD(o)$   
Meaning  $s$  has already read something in  $o$ 's dataset
  2. For all  $o' \in O$ ,  $o' \in PR(s) \Rightarrow COI(o') \neq COI(o)$   
Meaning  $s$  has not read any objects in  $o$ 's conflict of interest class
- Ignores sanitized data
- Initially,  $PR(s) = \emptyset$ , so initial read request granted

# Sanitizing

- Public information may belong to a CD
  - As it's publicly available, no conflicts of interest arise
  - So, should not affect ability of analysts to read
  - Typically, all sensitive data removed from such information before it is released publicly (called *sanitizing*)
- Add third condition to CW-Simple Security Condition:
  3.  $o$  is a sanitized object

# Writing

- Anthony, Susan work in same trading house
- Anthony can read BofA's CD, ARCO's CD
- Susan can read Citi's CD, ARCO's CD
- If Anthony could write to ARCO's CD, Susan can read it
  - Hence, indirectly, she can read information from BofA's CD, a conflict of interest

# CW<sup>\*</sup>-Property

- $s$  can write to  $o$  iff both of the following hold:
  1. The CW-simple security condition permits  $s$  to read  $o$ ; and
  2. For all *unsanitized* objects  $o'$ , if  $s$  can read  $o'$ , then  $CD(o') = CD(o)$
- Says that  $s$  can write to an object if all the (unsanitized) objects it can read are in the same dataset

# Multilevel Security (MLS)

A class of system that has system resources (particularly stored information) at more than one security level (*i.e.*, has different types of sensitive resources) and that permits concurrent access by users who differ in security clearance and need-to-know, but is able to prevent each user from accessing resources for which the user lacks authorization.

Such a system implements BLP or something like it.

# MLS Security for Role-Based Access Control

- Role based access control (RBAC) can implement BLP MLS rules given:
  - security constraints on users
  - constraints on read/write permissions
  - read and write level role access definitions
  - constraint on user-role assignments

# MLS Database Security

Department Table - U		
Did	Name	Mgr
4	accts	Cathy
8	PR	James

Employee - R			
Name	Did	Salary	Eid
Andy	4	43K	2345
Calvin	4	35K	5088
Cathy	4	48K	7712
James	8	55K	9664
Ziggy	8	67K	3054

(a) Classified by table

Department Table		
Did -U	Name -U	Mgr -R
4	accts	Cathy
8	PR	James

Employee			
Name -U	Did -U	Salary -R	Eid -U
Andy	4	43K	2345
Calvin	4	35K	5088
Cathy	4	48K	7712
James	8	55K	9664
Ziggy	8	67K	3054

(b) Classified by column (attribute)



# MLS Database Security

Department Table			
Did	Name	Mgr	
4	accts	Cathy	R
8	PR	James	U

Employee				
Name	Did	Salary	Eid	
Andy	4	43K	2345	U
Calvin	4	35K	5088	U
Cathy	4	48K	7712	U
James	8	55K	9664	R
Ziggy	8	67K	3054	R

(c) Classified by row (tuple)

Department Table		
Did	Name	Mgr
4 - U	accts - U	Cathy - R
8 - U	PR - U	James - R

Employee			
Name	Did	Salary	Eid
Andy - U	4 - U	43K - U	2345 - U
Calvin - U	4 - U	35K - U	5088 - U
Cathy - U	4 - U	48K - U	7712 - U
James - U	8 - U	55K - R	9664 - U
Ziggy - U	8 - U	67K - R	3054 - U

(d) Classified by element

# MLS Database Security

## Read Access

- DBMS enforces simple security rule (no read up)
- easy if granularity is entire database / table level
- inference problems if there is row granularity
  - if can query on restricted data can infer its existence
    - `SELECT Ename FROM Employee WHERE Salary > 50K`
  - solution is to check access to all query data
- ~~also have problems if have row granularity~~
  - null response indicates restricted/empty result
- no extra concerns if have element granularity

# Trusted Platform Module (TPM)

- Concept from Trusted Computing Group
- Hardware module at heart of hardware / software approach to trusted computing
- Uses a TPM chip:
  - motherboard, smart card, processor
  - working with approved hardware / software
  - generating and using crypto keys
- has 3 basic services: authenticated boot, certification, and encryption

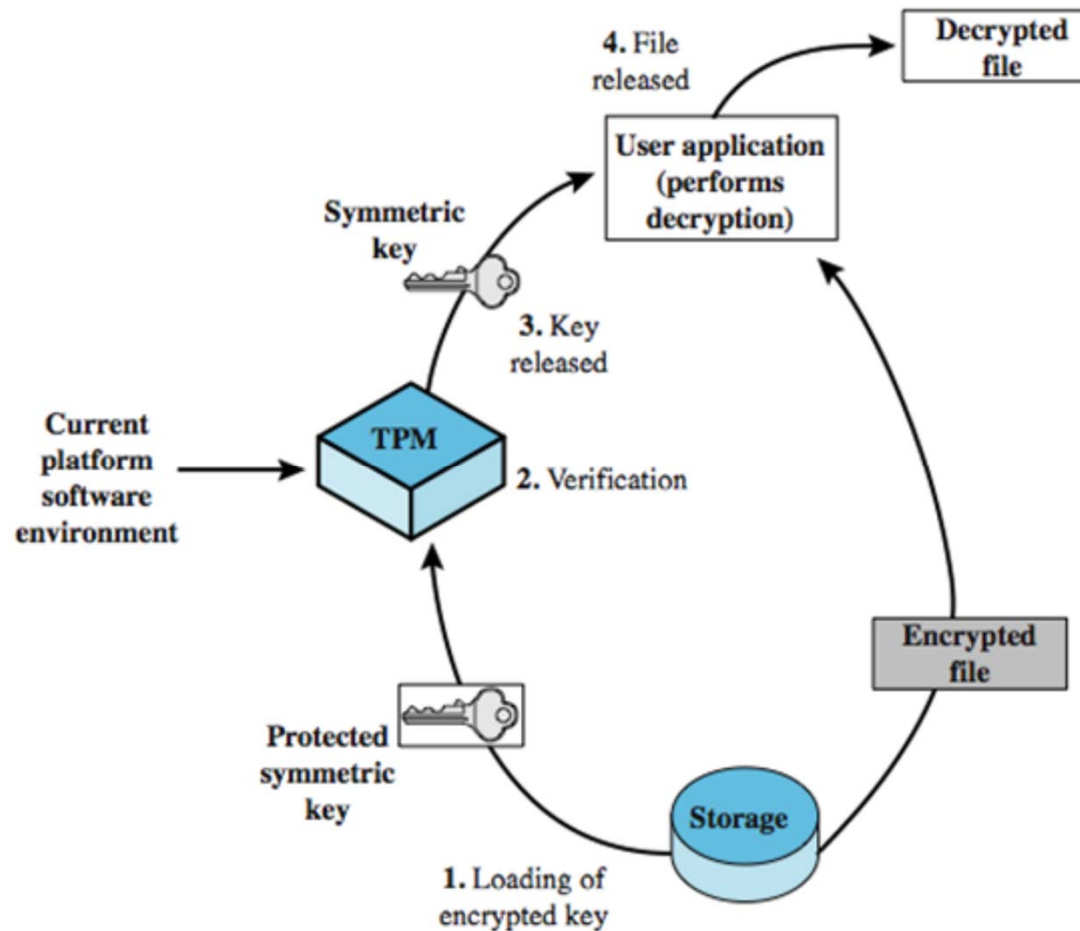
# Authenticated Boot Service

- Responsible for booting entire O/S in stages
- Ensuring each is valid and approved for use
  - verifying digital signature associated with code
  - keeping a tamper-evident log
- A log records versions of all code running
- Can then expand trust boundary
  - TPM verifies any additional software requested
    - confirms signed and not revoked
- Hence, we know the resulting configuration is well-defined with approved components

# Encryption Service

- Encrypts data so it can be decrypted
  - by a certain machine in given configuration
- Depends on
  - master secret key unique to machine
  - used to generate secret encryption key for every possible configuration, only usable in it
- Can also extend this scheme upward
  - create application key for desired application version running on desired system version

# Protected Storage Function



# Trusted Systems

- Security models aimed at enhancing trust
- Work started in early 1970's leading to:
  - Trusted Computer System Evaluation Criteria (TCSEC), Orange Book, in early 1980s
  - further work by other countries
  - resulting in Common Criteria in late 1990s
- Also Computer Security Center in NSA
  - with Commercial Product Evaluation Program
  - evaluates commercially available products
  - required for Defense use, freely published

# Common Criteria (CC)

- ISO standard for security requirements and defining evaluation criteria to give:
  - greater confidence in IT product security
  - from formal actions during process of:
    - development using secure requirements
    - evaluation confirming meets requirements
    - operation in accordance with requirements
- Specifies functional and assurance requirements.
- Evaluated products are listed for use



# CC Requirements

- Have a common set of potential security requirements for use in evaluation
- The “target of evaluation” (TOE) refers product / system subject to evaluation
- Functional requirements: define desired security behavior
- Assurance requirements: mechanisms to show that security measures effective and correct

# Example: Smartcard Protection Profile

- Simple protection profile example
- describes IT security requirements for smart card use by sensitive applications
- lists threats
- PP requirements:
  - 42 TOE security functional requirements
  - 24 TOE security assurance requirements
  - IT environment security requirements
- with rationale for selection

# Assurance

- “Degree of confidence that the security controls operate correctly and protect the system as intended”
- Applies to:
  - product security requirements, security policy, product design, implementation, operation
- Various approaches analyzing, checking, testing various aspects

# Evaluation Assurance Levels

- EAL 1 - functionally tested
- EAL 2: structurally tested
- EAL 3: methodically tested and checked
- EAL 4: methodically designed, tested, and reviewed
- EAL 5: semiformally designed and tested
- EAL 6: semiformally verified design and tested
- EAL 7: formally verified design and tested

# Evaluation

- ensure security features correct & effective
- performed during / after TOE development
- higher levels need greater rigor and cost
- input: security target, evidence, actual TOE
- result: confirm security target satisfied for TOE
- process relates security target to some of TOE:
  - high-level design, low-level design, functional spec, source code, object code, hardware realization
- higher levels need semiformal / formal models

# Evaluation Parties and Phases

- evaluation parties:
  - sponsor - customer or vendor
  - developer - provides evidence for evaluation
  - evaluator - confirms requirements satisfied
  - certifier - agency monitoring evaluation process
- phases:
  - preparation, conduct of evaluation, conclusion
- government agency regulates, *e.g.* US CCEVS
- have peering agreements between countries
  - saving time / expense by sharing results

# Questions

