CS 591: Introduction to Computer Security

Lecture 5: Integrity Models

James Hook
(Some materials from Bishop, copyright 2004)

Objectives

- Integrity models in context
- Introduce integrity models
- Begin hybrid models

Plumbing Analogy

- Potable water
  - Cold
  - Hot
- Storm water
- Gray water
- Brown water

- Shower
- Toilet
- Washing machine
- The “CSO” problem

- What comes out of the tap?

Simple integrity

- Integrity Levels
  - Potable water
    - Cold
    - Hot
  - Storm water
  - Gray water
  - Brown water

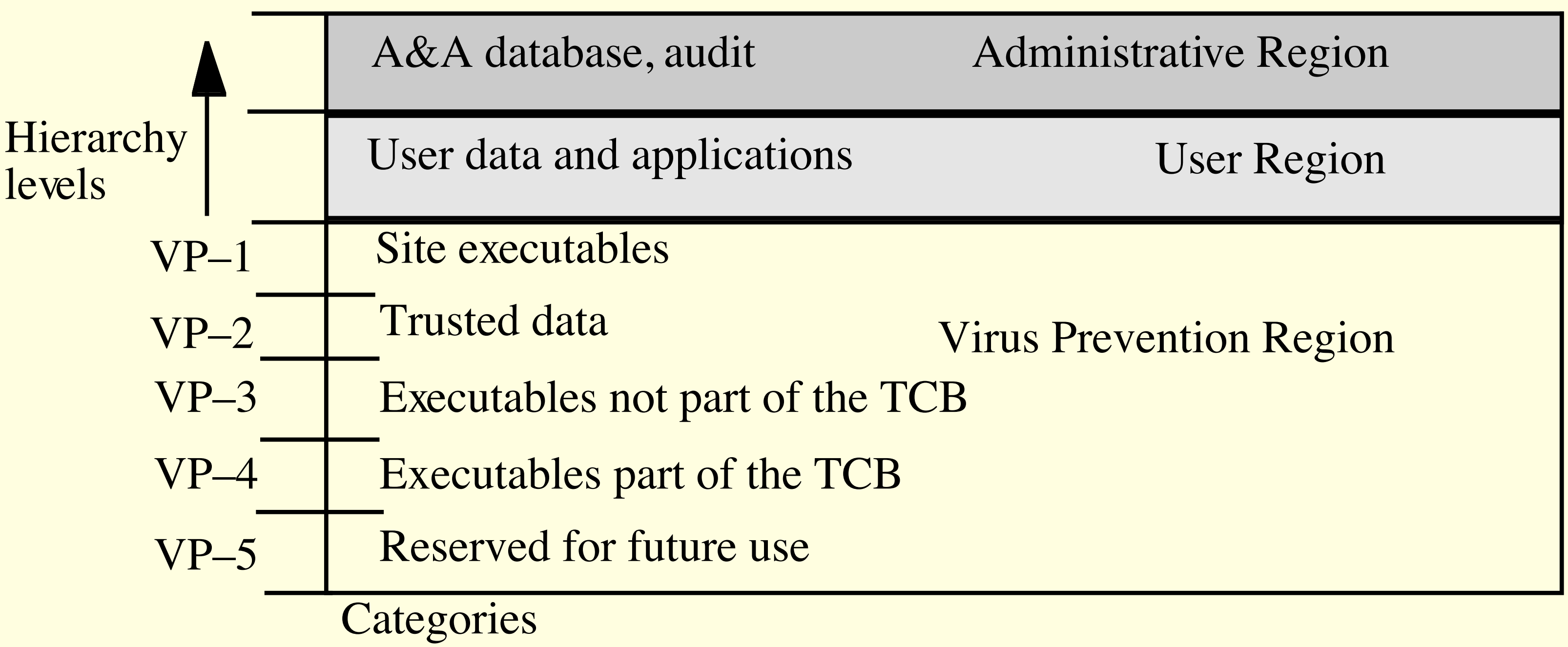
- Multilevel devices:
  - Shower
  - Toilet
  - Washing machine

- What kind(s) of water can people easily obtain (read/execute)?
- What kind(s) of water can people produce (write)?

Last Lecture

- Bell LaPadula Confidentiality
- Lattice of security levels
  - No read up
  - No write down
- DG/UX realization of Bell LaPadula

MAC Regions



IMPL_HI is "maximum" (least upper bound) of all levels

IMPL_LO is "minimum" (greatest lower bound) of all levels

Integrity

- Extrinsic: A system has integrity if it is trusted
- Integrity is not just a property of the information system
- A perfect information system could lose integrity in a corrupt organization

Integrity Principles

- Separation of Duty:
  - If two or more steps are required to perform a critical function, at least two different people should perform the steps

Integrity Principles

- Separation of Duty
- Separation of Function
  - Developers do not develop new programs on productions systems

Integrity Principles

- Separation of Duty
- Separation of Function
- Auditing
  - Record what actions took place and who performed them
  - Contributes to both recovery and accountability

Chapter 6: Integrity Policies

- Overview
- Requirements
- Biba's models
- Clark-Wilson model

Overview

- Requirements
  - Very different than confidentiality policies
- Biba's model
- Clark-Wilson model

Requirements of Policies

1. Users will not write their own programs, but will use existing production programs and databases.
2. Programmers will develop and test programs on a non-production system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
3. A special process must be followed to install a program from the development system onto the production system.
4. The special process in requirement 3 must be controlled and audited.
5. The managers and auditors must have access to both the system state and the system logs that are generated.

Biba Integrity Model

- Set of subjects S, objects O, integrity levels I, relation $\leq \subseteq I \times I$ holding when second dominates first
- *min*: $I \times I \rightarrow I$ returns lesser of integrity levels
- *i*: $S \cup O \rightarrow I$ gives integrity level of entity
- $\underline{\text{r}}$: $S \times O$ means $s \in S$ can read $o \in O$
- $\underline{\text{w}}$, $\underline{\text{x}}$ defined similarly

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 relationship between integrity and trustworthiness
- Important point: *integrity levels are **not** security levels*

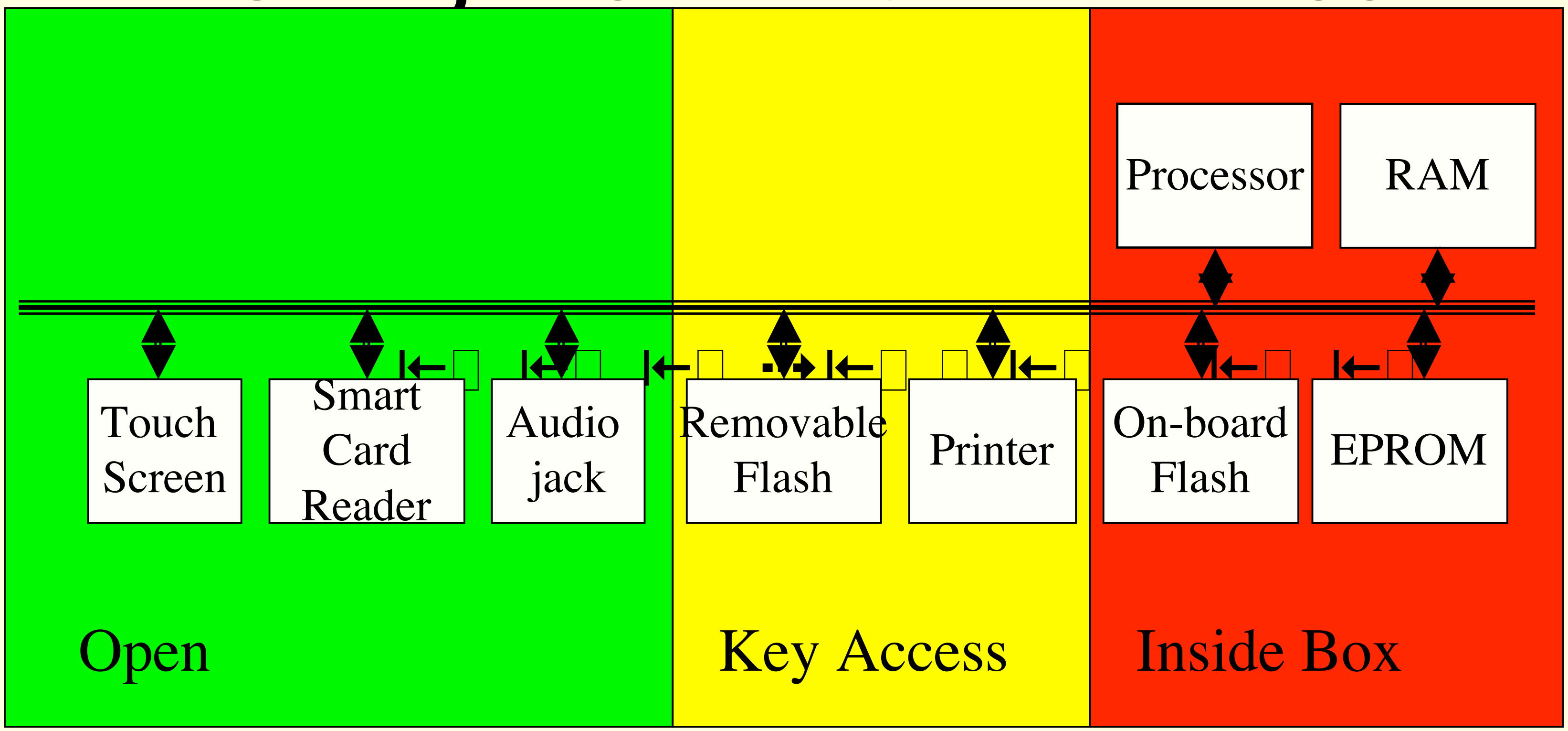
Biba’s Model

- Similar to Bell-LaPadula model
    1. $s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$
    2. $s \in S$ can write to $o \in O$ iff $i(o) \leq i(s)$
    3. $s_1 \in S$ can execute $s_2 \in S$ iff $i(s_2) \leq i(s_1)$
- Add compartments and discretionary controls to get full dual of Bell-LaPadula model
- Information flow result holds
    - Different proof, though
- Actually the “strict integrity model” of Biba’s set of models

LOCUS and Biba

- Goal: prevent untrusted software from altering data or other software
- Approach: make levels of trust explicit
  - *credibility rating* based on estimate of software's trustworthiness (0 untrusted, *n* highly trusted)
  - *trusted file systems* contain software with a single credibility level
  - Process has *risk level* or highest credibility level at which process can execute
  - Must use *run-untrusted* command to run software at lower credibility level

Voting Machine with Biba



- Subjects? Objects? Integrity Levels?

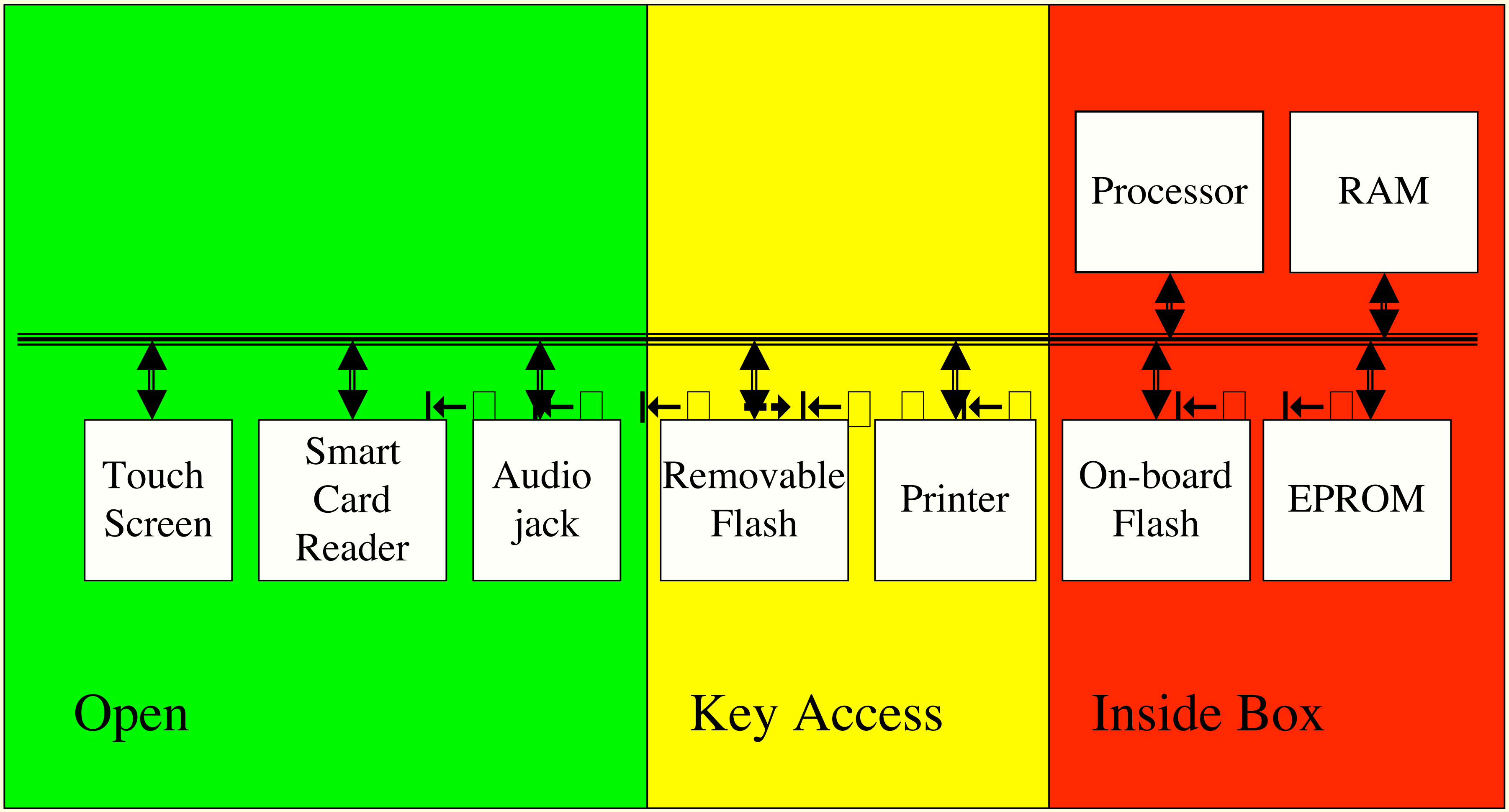
Example

- Elaborate the Biba integrity model for this system by assigning integrity levels to all key files. Specifically assign integrity levels for creating or modifying these files.
- Several known exploits of the system rely on infection via removable media. Propose a mechanism that uses the trusted authentication mechanism and integrity model to prevent these exploits.

Example (cont)

- Argue that the intended operations can be carried out by appropriate subjects without violating the policy.
- Argue that with these mechanisms and a faithful implementation of the integrity model that Felten's vote stealing and denial of service attacks would not be allowed.

Voting Machine Architecture



Boot Process

- Boot device specified by hardware jumpers (inside box)
  - EPROM
  - on-board flash (default)
  - ext flash
- On Boot:
  - Copy bootloader into RAM; init hardware
  - Scan Removable flash for special files
    - "fboot.nb0" => replace bootloader in on-board flash
    - "nk.bin" => replace OS in on-board flash
    - "EraseFFX.bsq" => erase file system on on-board flash
  - If no special files uncompress OS image
  - Jump to entry point of OS

Boot (continued)

- On OS start up:
  - run Filesys.exe
    - unpacks registry
    - runs programs in HKEY_LOCAL_MACHINE\Init
      - shell.exe (debug shell)
      - device.exe (Device manager)
      - gwes.exe (graphics and event)
      - taskman.exe (Task Manager)
  - Device.exe mounts file systems
    - \ (root): RAM only
    - \FFX: mount point for on-board flash
    - \Storage Card: mount point for removable flash

Boot (continued)

- Customized taskman.exe
  - Check removable flash
    - explorer.glb => launch windows explorer
    - *.ins => run proprietary scripts
      - (script language has buffer overflow vulnerabilities)
      - used to configure election data
    - default => launch “BallotStation”
      - \FFX\Bin\BallotStation.exe

BallotStation

- Four modes: pre-download, pre-election testing, election, post-election
- Mode recorded in election results file
  - \Storage Card\CurrentElection\election.brs

Stealing Votes

- Malicious processes runs in parallel with BallotStation
- Polls election results file every 15 seconds
  - If election mode and new results
    - temporarily suspend Ballot Station
    - steal votes
    - resume Ballot Station

Viral propagation

- Malicious bootloader
  - Infects host by replacing existing bootloader in on-board flash
  - subsequent bootloader updates print appropriate messages but do nothing
- fboot.nb0
  - package contains malicious boot loader
  - and vote stealing software

Discussion

- Having developed this design, it is now time to critique it!
  - Are you satisfied with the protection against external threats?
  - Are you satisfied with the protection against insider threats?

Clark-Wilson Integrity Model

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

Entities

- CDIs: constrained data items
  - Data subject to integrity controls
- UDIs: unconstrained data items
  - Data not subject to integrity controls
- IVPs: integrity verification procedures
  - Procedures that test the CDIs conform to the integrity constraints
- TPs: transaction procedures
  - Procedures that take the system from one valid state to another

Certification Rules 1 and 2

CR1 When any IVP is run, it must ensure all CDIs are in a valid state

CR2 For some associated set of CDIs, a TP must transform those CDIs in a valid state into a (possibly different) valid state

- Defines relation *certified* that associates a set of CDIs with a particular TP
- Example: TP balance, CDIs accounts, in bank example

Enforcement Rules 1 and 2

ER1 The system must maintain the certified relations and must ensure that only TPs certified to run on a CDI manipulate that CDI.

ER2 The system must associate a user with each TP and set of CDIs. The TP may access those CDIs on behalf of the associated user. The TP cannot access that CDI on behalf of a user not associated with that TP and CDI.

- System must maintain, enforce certified relation
- System must also restrict access based on user ID (*allowed* relation)

Users and Rules

CR3 The allowed relations must meet the requirements imposed by the principle of separation of duty.

ER3 The system must authenticate each user attempting to execute a TP

- Type of authentication undefined, and depends on the instantiation
- Authentication *not* required before use of the system, but *is* required before manipulation of CDIs (requires using TPs)

Logging

CR4 All TPs must append enough information to reconstruct the operation to an append-only CDI.

- This CDI is the log
- Auditor needs to be able to determine what happened during reviews of transactions

Handling Untrusted Input

CR5 Any TP that takes as input a UDI may perform only valid transformations, or no transformations, for all possible values of the UDI. The transformation either rejects the UDI or transforms it into a CDI.

- In bank, numbers entered at keyboard are UDIs, so cannot be input to TPs. TPs must validate numbers (to make them a CDI) before using them; if validation fails, TP rejects UDI

Separation of Duty In Model

ER4 Only the certifier of a TP may change the list of entities associated with that TP. No certifier of a TP, or of an entity associated with that TP, may ever have execute permission with respect to that entity.

- Enforces separation of duty with respect to certified and allowed relations

Comparison With Requirements

1. Users can’t certify TPs, so CR5 and ER4 enforce this
2. Procedural, so model doesn’t directly cover it; but special process corresponds to using TP
   - No technical controls can prevent programmer from developing program on production system; usual control is to delete software tools
3. TP does the installation, trusted personnel do certification

Comparison With Requirements

4. CR4 provides logging; ER3 authenticates trusted personnel doing installation; CR5, ER4 control installation procedure
   - New program UDI before certification, CDI (and TP) after
5. Log is CDI, so appropriate TP can provide managers, auditors access
   - Access to state handled similarly

Comparison to Biba

- Biba
  - No notion of certification rules; trusted subjects ensure actions obey rules
  - Untrusted data examined before being made trusted
- Clark-Wilson
  - Explicit requirements that *actions* must meet
  - Trusted entity must certify *method* to upgrade untrusted data (and not certify the data itself)

Key Points

- Integrity policies deal with trust
  - As trust is hard to quantify, these policies are hard to evaluate completely
  - Look for assumptions and trusted users to find possible weak points in their implementation
- Biba based on multilevel integrity
- Clark-Wilson focuses on separation of duty and transactions

Presentation

- Bishop Chapter 7 slides