CSCE48503: Information Security

Week 4: Security Policies

University of Arkansas

Feb 3, 2025

UNIVERSITY OF ARKANSAS

Schedule [Tentative]

RIOT LAB

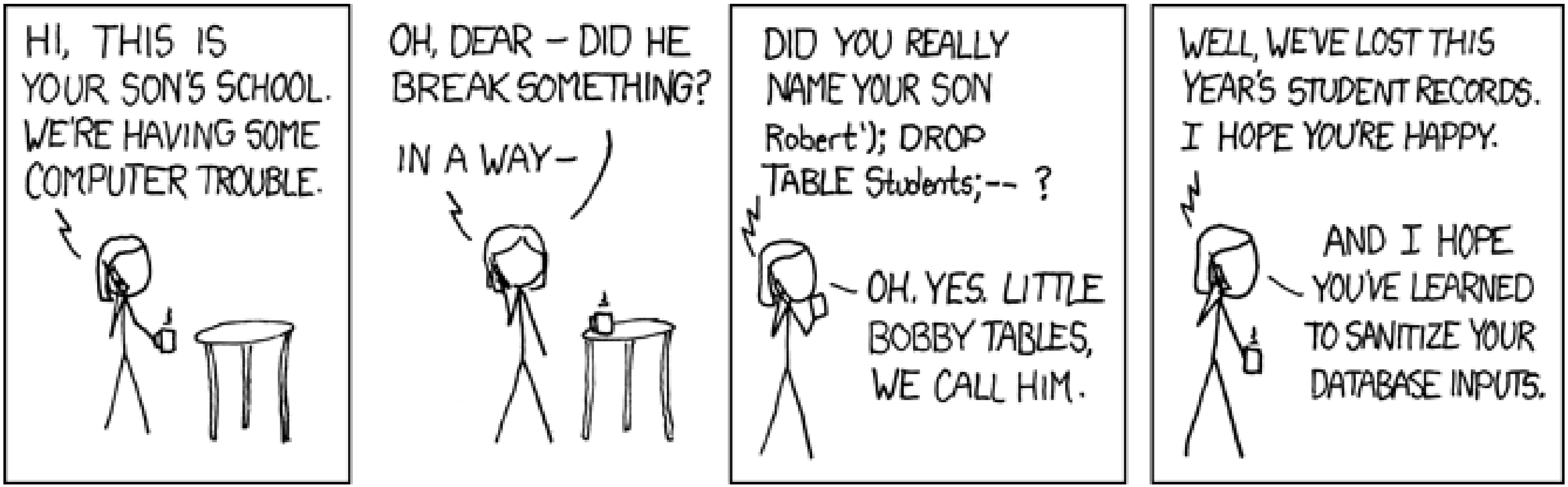
Week 1: Intro, Syllabus, CIA (Expectations)	[13Jan2025]	
Week 2: Security Basics	[20Jan2025]	(MLK Holiday)
Week 3: Access Control	[27Jan2025]	
Week 4: Security Policies (Week 1)	[3Feb2025]	
Week 5: Security Policies (Week 2)	[10Feb2025]	(S4x25 Conf)
Week 6: Cryptography Basics (Week 1)	[17Feb2025]	
Week 7: Cryptography Basics (Week 2)	[24Feb2025]	
Week 8: Cryptography Basics (Week 3)	[3Mar2025]	(Maj Lang)
Week 9: Mid-Term Review and Test	[10Mar2025]	
Week 10: Operating Systems Security & Malware	[17Mar2025]	
Week 11: Spring Break! (Be Safe)	[24Mar2025]	(Spring Break)
Week 12: Network Security (Week 1)	[31Mar2025]	
Week 13: Network Security (Week 2)	[7Apr2025]	(IEEE DC)
Week 14: Web Security	[14Apr2025]	
Week 15: Advanced Topics	[21Apr2025]	
Week 16: FINAL Review	[28Apr2025]	
Week 17: FINAL Exam Respondus and in Classroom	[7May2025 @ 10:15am]	

UNIVERSITY OF ARKANSAS

ACM

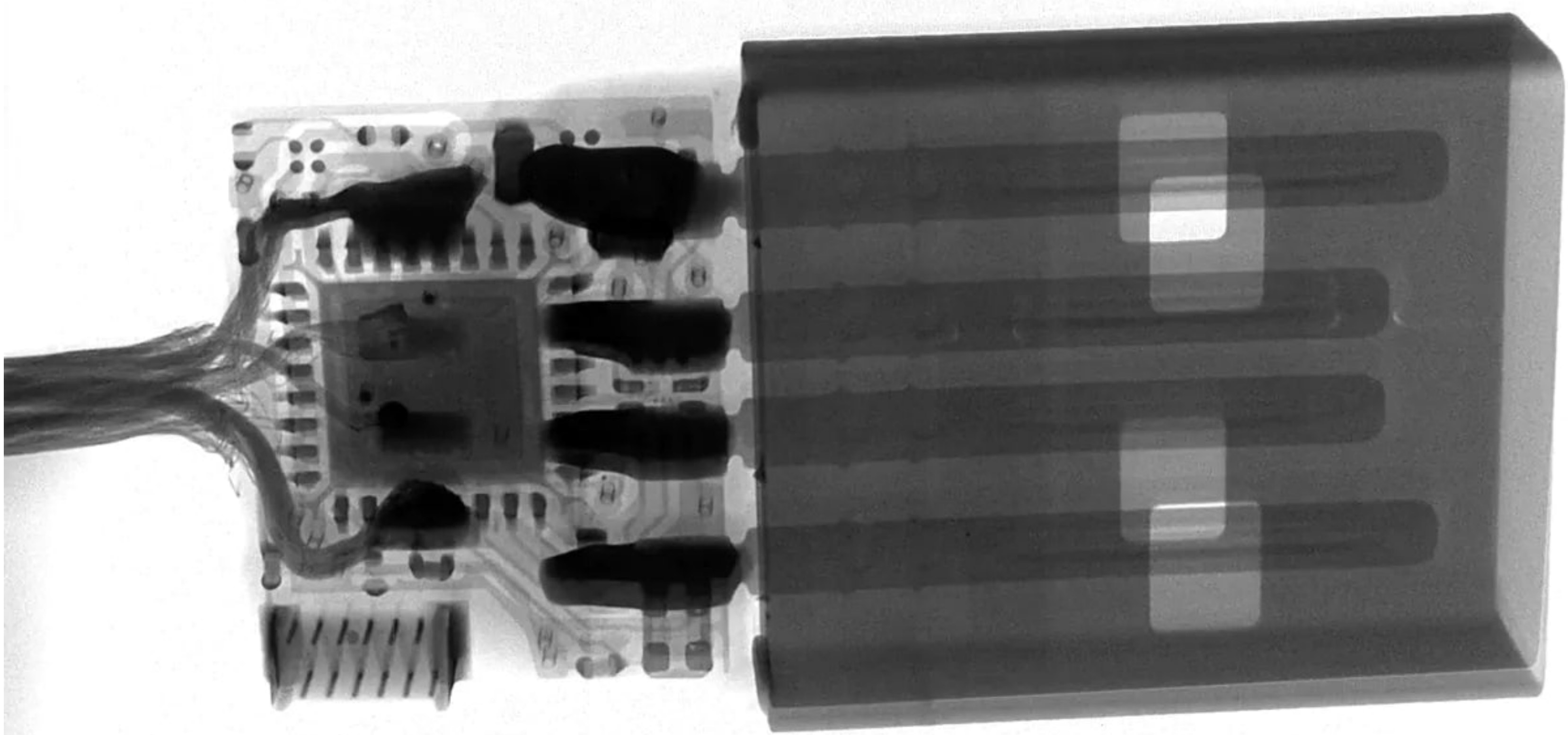
RIOT LAB

OWASP Example



https://xkcd.com/327/

Bad USB Example



Bad USB Example

Confidentiality Policy

- **Goal: prevent the unauthorized disclosure of information**
  - **Deals with information flow**
  - **Integrity incidental**
- **Multi-level security models are best-known examples**
  - **Bell-LaPadula Model basis for many, or most, of these**

Bell-LaPadula Model, Step 1

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist of *security clearance* $L(s)$
  - Objects have *security classification* $L(o)$

Example

security level	*subject*	*object*
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

- **Tamara can read all files**
- **Claire cannot read Personnel or E-Mail Files**
- **Ulaley can only read Telephone Lists**

Reading Information

- **Information flows up, not down**
  - **"Reads up" disallowed, "reads down" allowed**
    - **A subject can read all documents at or below his level of security, but cannot read any documents above his level of security**
    - **Prevents learning secrets at a higher security level**
- **Simple Security Condition (Step 1)**
  - **Subject s can read object o iff $L(o) \le L(s)$ and s has permission to read o**
    - **Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)**
  - **Sometimes called "no reads up" rule**

Writing Information

- **Information flows up, not down**
  - **"Writes up" allowed, "writes down" disallowed**
    - **A subject can write documents at or above his level of security, but cannot write documents below his level**
    - **Prevents leaks of secrets**
- ***-Property (Step 1)**
  - **Subject *s* can write object *o* iff $L(s) \le L(o)$ and *s* has permission to write *o***
    - **Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)**
  - **Sometimes called "no writes down" rule**

Basic Security Theorem, Step 1

- **If a system is initially in a secure state, and every transition of the system satisfies the simple security condition, step 1, and the *-property, step 1, then every state of the system is secure**
  - **Proof: induct on the number of transitions**

Bell-LaPadula Model, Step 2

- **Expand notion of security level to include categories**
- **Each category describes a kind of information**
- **Security level is (*clearance*, *category set*)**
- **Examples**
  - **(Top Secret, { NUC, EUR, ASI })**
  - **(Confidential, { EUR, ASI })**
  - **(Secret, { NUC, ASI })**

Levels

- (A, C) *dom* (A', C') iff $A' \leq A$ and $C' \subseteq C$
- Examples
  - (Top Secret, {NUC, ASI}) *dom* (Secret, {NUC})
  - (Secret, {NUC, EUR}) *dom* (Confidential,{NUC, EUR})
  - (Top Secret, {NUC}) $\neg$*dom* (Confidential, {EUR})

Levels and Ordering

- **Security levels partially ordered**
  - **Any pair of security levels may (or may not) be related by *dom***
- **"dominates" serves the role of "greater than" in step 1**

Reading Information

- **Information flows up, not down**
  - **"Reads up" disallowed, "reads down" allowed**
    - **A subject can read all documents at or below his level of security, but cannot read any documents above his level of security**
    - **Prevents learning secrets at a higher security level**
- **Simple Security Condition (Step 2)**
  - **Subject *s* can read object *o* iff *L*(*s*) *dom* *L*(*o*) and *s* has permission to read *o***
    - **Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)**
  - **Sometimes called "no reads up" rule**

Writing Information

- **Information flows up, not down**
  - **"Writes up" allowed, "writes down" disallowed**
    - **A subject can write documents at or above his level of security, but cannot write documents below his level**
    - **Prevents leaks of secrets**
- ***-Property (Step 2)**
  - **Subject *s* can write object *o* iff *L*(*o*) *dom* *L*(*s*) and *s* has permission to write *o***
    - **Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)**
  - **Sometimes called "no writes down" rule**

Problem

- **At times, a subject must communicate with another subject at a lower level. This requires the higher-level subject to write into a lower-level object that the lower-level subject can read**

Problem

- **Colonel has (Secret, {NUC, EUR}) clearance, Major has (Secret, {EUR}) clearance**
  - **The Colonel needs to send a message to the major, i.e., write a document that has at most the (SECRET, { EUR }) classification**

Allowed?

No! (SECRET, { NUC, EUR }) *dom* (SECRET, { EUR }), so the *-property is violated

Major can talk to colonel ("write up" or "read down")
Colonel cannot talk to major ("read up" or "write down")

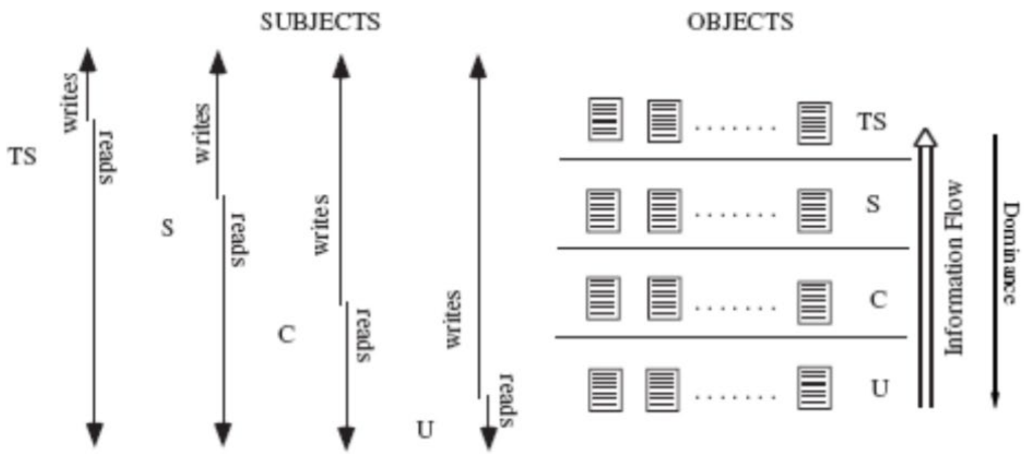
Clearly absurd!

Solution

- **Define maximum, current levels for subjects**
  - *maxlevel*(*s*) *dom curlevel*(*s*)
- **Example**
  - **Treat Major as an object (Colonel is writing to him/her)**
  - **Colonel has *maxlevel* (Secret, { NUC, EUR })**
  - **Colonel sets *curlevel* to (Secret, { EUR })**
  - **Now *L*(Major) *dom curlevel*(Colonel)**
    - **Colonel can write to Major without violating "no writes down"**
  - **Does *L*(*s*) mean *curlevel*(*s*) or *maxlevel*(*s*)?**
    - **Formally, we need a more precise notation**

Key Points

- **Confidentiality models restrict flow of information**
- **Uses No Read Up & No Write Down**



- **Bell-LaPadula models multilevel security**
  - **Subject at a high level may not convey info to a subject at a non-comparable level**
  - **Cornerstone of much work in computer security**

Reading

- Chapter 9.2-9.2.1

Integrity Policy

- 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.**

Principles of Operation

- *Separation of duty*: **if two or more steps are required to perform a critical function, at least two different people should perform the steps**
- *Separation of function*: **different entities should perform different functions**
- *Auditing*: **recording enough information to ensure the abilities to both recover and determine accountability**

Biba Integrity Model

- Designed to guard against the unauthorized modification of data

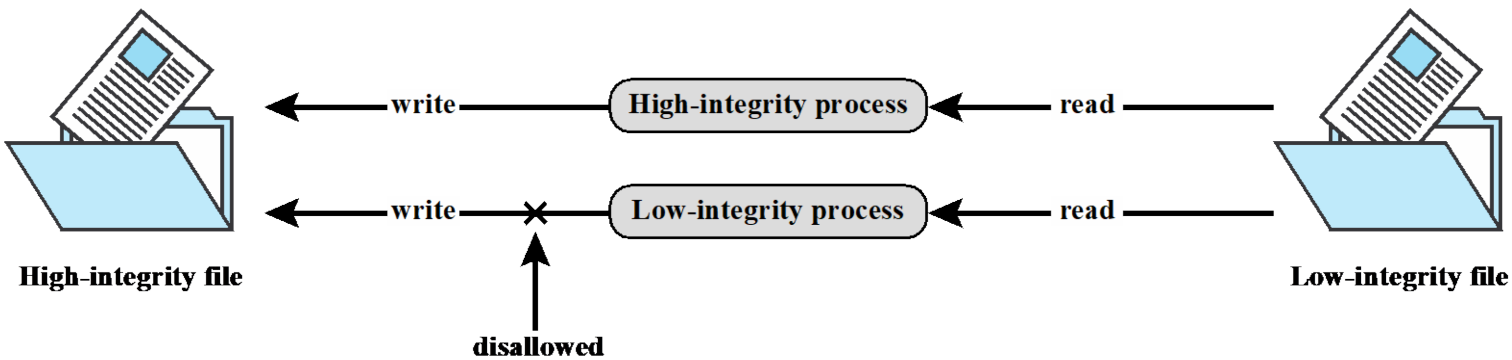


Figure 13.4 Contamination With Simple Integrity Controls [GASS88]

Biba Integrity Model

- Similar to Bell-LaPadula model
- Strict integrity policy:
  1. $s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$
     - Can only read up (so high level but compromised subjects cannot copy low integrity data up)
     - integrity confinement
  2. $s \in S$ can write to $o \in O$ iff $i(o) \leq i(s)$
     - Can only write down, so can't contaminate high-level data
     - simple integrity
  3. $s_1 \in S$ can execute $s_2 \in S$ iff $i(s_2) \leq i(s_1)$
     - only want to allow communication to go "down"
- Add compartments and discretionary controls to get full dual of Bell-LaPadula 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 relationship between integrity and trustworthiness**
- **Important point:** *integrity levels are not security levels*
  - **Security levels primarily limit the flow of information**
  - **Integrity levels primarily inhibit the modification of information**

Comparison to BLP

- Both are designed for the military, to protect high-level secrets
- If you need to protect secrets, use Bell-LaPadula
  - No Write Down
  - No Read Up
- If you need to stay on target, use Biba
  - No Write Up
  - No Read Down

Clark-Wilson Integrity Model

- **Designed for businesses, to protect the integrity of data at all levels, not just the high value secrets**
- **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/Consistency property: $TB = D + YB - W$**
- **Basic operation: Transactions**
  - ***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: (balance, account1), (balance, account2), …, (balance, account*n*) ∈ *C*, *C* be the certified relation**

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

Two concepts

- Well-formed transactions: a user can manipulate data in constrained ways
- Separation of duty: one can create a transaction but not execute it

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**

You should know

- Know that Biba and Clark-Wilson models are integrity models
- What are the rules of Biba model for read and write? Not in formula but in plain language.
- The concept of well-formed transaction in Clark-Wilson model
- The concept of separation-of-duty

Reading

- **Chapter 9.2.2**

Hybrid Policies

- **Overview**
  - **Addresses integrity and confidentiality**
- **Chinese Wall Model**
  - **Focuses on conflict of interest**
- **RBAC**
  - **Base controls on job function**

Chinese Wall Model

Problem:

- Tony advises Bank of America about investments
- He is asked to advise Citibank about investments

- Conflict of interest to accept, because his advice for either bank would affect his advice to the other bank

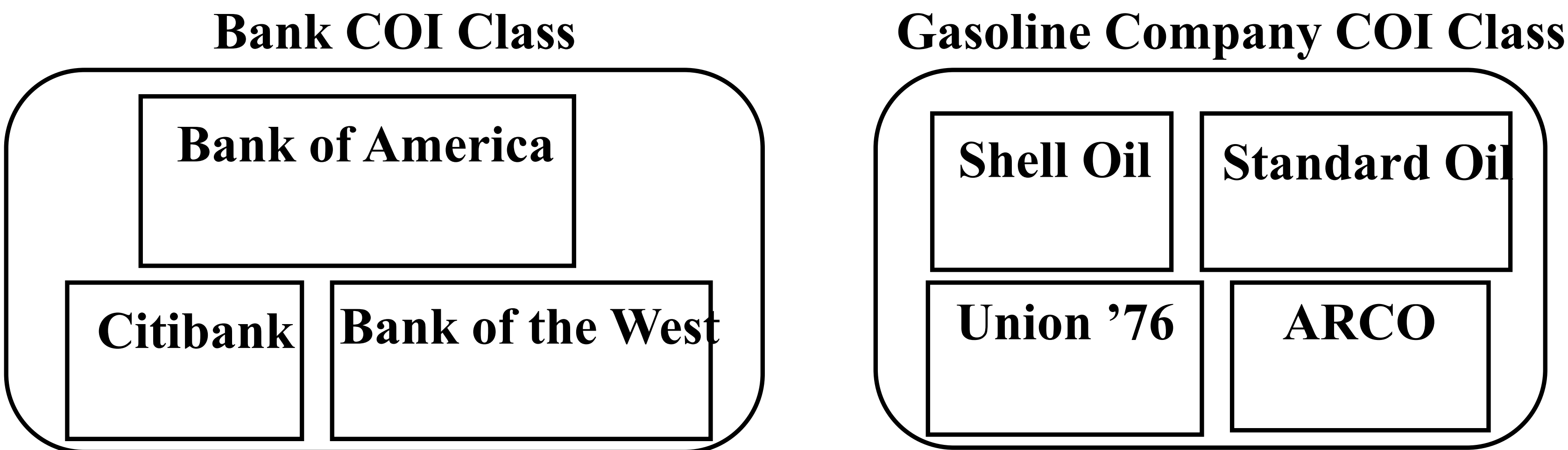
Organization

- 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 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*)**
  - **Assume: each object belongs to exactly one *COI* class**

Example



- Can access Citibank's CD and ARCO's CD
- Cannot access Citibank's CD and Bank of America's CD

Temporal Element

- **If Anthony reads any CD in a COI, he can *never* read another CD in that COI**
  - **Anthony first worked on Bank of America and was then transferred to Citibank**
  - **Possible that information learned earlier may allow him to make decisions later**

CW-Simple Security Condition

- Let $PR(S)$ be set of objects that S has already read
- 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 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
- Initially, $PR(s) = \varnothing$, so initial read request granted
- Ignores sanitized data (see below)

Sanitization

- **Public information may belong to a CD**
  - **As is 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 *sanitization*)**
- **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 Bank 1's CD, Gas' CD
- Susan can read Bank 2's CD, Gas' CD
- If Anthony could write to Gas' CD, Susan can read it
  - Hence, indirectly, she can read information from Bank 1's CD, a clear conflict of interest

CW-*-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

RBAC: Role-based Access Control

- **Access depends on function, not identity**
  - **Example:**
    - **Allison, bookkeeper for Math Dept, has access to financial records.**
    - **She leaves.**
    - **Betty hired as the new bookkeeper, so she now has access to those records**
  - **The role of "bookkeeper" dictates access, not the identity of the individual.**

Definitions

- **Role *r*: collection of job functions**
  - ***trans*(*r*): set of authorized transactions for *r***
- **Active role of subject *s*: role *s* is currently in**
  - ***actr*(*s*)**
- **Authorized roles of a subject *s*: set of roles *s* is authorized to assume**
  - ***authr*(*s*)**
- ***canexec*(*s*, *t*) iff subject *s* can execute transaction *t* at current time**

Axioms

- Let S be the set of subjects and T the set of transactions.
- *Rule of role assignment*:

$$(\forall s \in S)(\forall t \in T)\ [canexec(s, t) \rightarrow actr(s) \neq \varnothing].$$

  - If *s* can execute a transaction, it has a role
  - This ties transactions to roles
- *Rule of role authorization*:

$$(\forall s \in S)\ [actr(s) \subseteq authr(s)].$$

  - Subject must be authorized to assume an active role (otherwise, any subject could assume any role)

Axiom

- *Rule of transaction authorization*:

$$(\forall s \in S)(\forall t \in T)$$

$$[canexec(s, t) \rightarrow t \in trans(actr(s))].$$

  - **If a subject *s* can execute a transaction, then the transaction is an authorized one for the role *s* has assumed**

Containment of Roles

- **Trainer can do all transactions that trainee can do (and then some). This means role *r* contains role *r*′ (*r* > *r*′). So:**

$$(\forall s \in S)[\, r' \in authr(s) \wedge r > r' \rightarrow r \in authr(s)\,]$$

Separation of Duty

- **Same individual cannot assume both roles**
- **Let *r* be a role, and let *s* be a subject such that $r \in auth(s)$. Then the predicate *meauth*(*r*) (for mutually exclusive authorizations) is the set of roles that *s* cannot assume because of the separation of duty requirement.**
- **Separation of duty:**

$$(\forall r_1, r_2 \in R)\,[\, r_2 \in meauth(r_1) \rightarrow$$
$$[\,(\forall s \in S)\,[\, r_1 \in authr(s) \rightarrow r_2 \notin authr(s)\,]\,]\,]$$

Key Points

- **Hybrid policies deal with both confidentiality and integrity**
  - **Different combinations of these**
- **RBAC model controls access based on functionality**

Reading

- **Chapter 9.2.3**