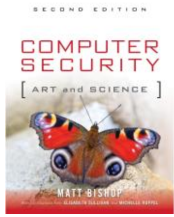
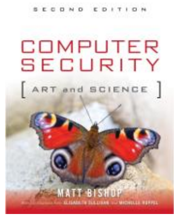


Hybrid Policies

Chapter 8



Overview

- Chinese Wall Model
  - Focuses on conflict of interest
- CISS Policy
  - Combines integrity and confidentiality
- ORCON
  - Combines mandatory, discretionary access controls
- RBAC
  - Base controls on job function

Chinese Wall Model

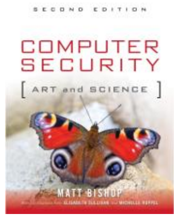
Problem:

- Tony advises American Bank about investments
- He is asked to advise Toyland Bank about investments

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

Organization

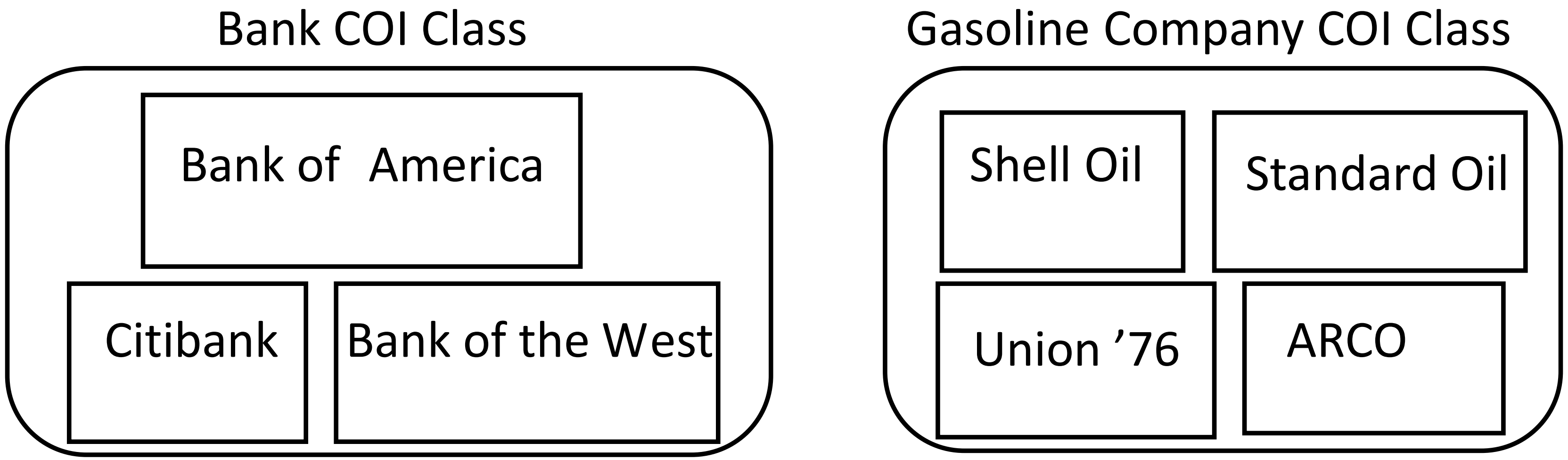
- Organize entities into "conflict of interest" classes
- Control subject accesses to each class
- Control writing to all classes to ensure information is not passed along in violation of rules
- Allow sanitized data to be viewed by everyone

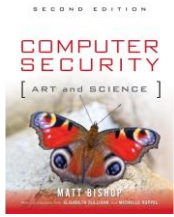


Definitions

- *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

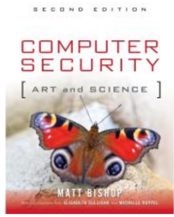
Example


Bank COI Class
Bank of America
Citibank
Bank of the West
Gasoline Company COI Class
Shell Oil
Standard Oil
Union ’76
ARCO



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



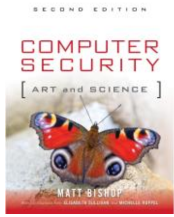
CW-Simple Security Condition

- *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 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 (see below)
- Initially, $PR(s) = \varnothing$, so initial read request granted

Sanitization

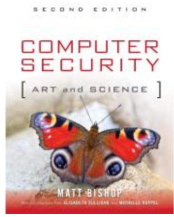
- Public information may belong to a CD
  - As is publicly available, no conflicts of interest arise
  - So, should not affect ability of analysts to read
  - Typically, all sensitive data removed from such information before it is released publicly (called *sanitization*)
- Add third condition to CW-Simple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

  3. *o* is a sanitized object



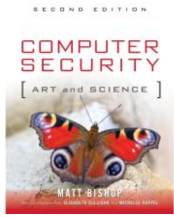
Writing

- 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



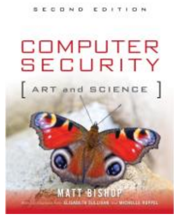
CW-*-Property

- *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



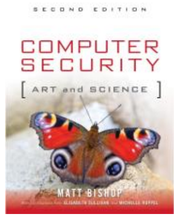
Formalism

- Goal: figure out how information flows around system
- S set of subjects, O set of objects, $L = C \times D$ set of labels
- $l_1: O \rightarrow C$ maps objects to their COI classes
- $l_2: O \rightarrow D$ maps objects to their CDs
- $H(s, o)$ true iff s has *or had* read access to o
- $R(s, o)$: s's request to read o



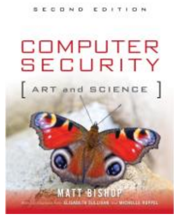
Axioms

- Axiom 8-1. For all o, $o' \in O$, if $l_2(o) = l_2(o')$, then $l_1(o) = l_1(o')$
  - CDs do not span COIs.
- Axiom 8-2. $s \in S$ can read $o \in O$ iff, for all $o' \in O$ such that $H(s, o')$, either $l_1(o') \neq l_1(o)$ or $l_2(o') = l_2(o)$
  - *s* can read *o* iff *o* is either in a different COI than every other *o′* that *s* has read, or in the same CD as *o*.



More Axioms

- Axiom 8-3. $\neg H(s, o)$ for all $s \in S$ and $o \in O$ is an initially secure state
    - Description of the initial state, assumed secure
- Axiom 8-4. If for some $s \in S$ and for all $o \in O$, $\neg H(s, o)$, then any request $R(s, o)$ is granted
    - If s has read no object, it can read any object



Which Objects Can Be Read?

Theorem 8-1: Suppose $s \in S$ has read $o \in O$. If s can read $o' \in O$, $o' \neq o$, then $l_1(o') \neq l_1(o)$ or $l_2(o') = l_2(o)$.

- Says *s* can read only the objects in a single CD within any COI

Proof

Assume false. Then

$$H(s, o) \wedge H(s, o') \wedge l_1(o') = l_1(o) \wedge l_2(o') \neq l_2(o)$$

Assume s read o first. Then $H(s, o)$ when s read o, so by Axiom 8-2, $l_1(o') \neq l_1(o)$ or $l_2(o') = l_2(o)$, so

$$(l_1(o') \neq l_1(\mathrm{o}) \vee l_2(o') = l_2(o)) \wedge (l_1(o') = l_1(o) \wedge l_2(o') \neq l_2(o))$$

Rearranging terms,

$$(l_1(o') \neq l_1(o) \wedge l_2(o') \neq l_2(o) \wedge l_1(o') = l_1(o)) \vee (l_2(o') = l_2(o) \wedge l_2(o') \neq l_2(o) \wedge l_1(o') = l_1(o))$$

which is obviously false, contradiction.

Lemma

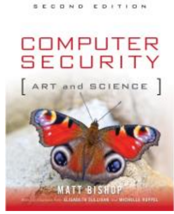
Lemma 8-2: Suppose a subject $s \in S$ can read an object $o \in O$. Then s can read no o' for which $l_1(o') = l_1(o)$ and $l_2(o') \neq l_2(o)$.

- So a subject can access at most one CD in each COI class
- Sketch of proof: Initial case follows from Axioms 8-3, 8-4. If $o' \neq o$, theorem immediately gives lemma.

COIs and Subjects

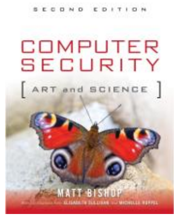
Theorem 8-2: Let $c \in C$. Suppose there are n objects $o_i \in O$, $1 \leq i \leq n$, such that $l_1(o_i) = c$ for $1 \leq i \leq n$, and $l_2(o_i) \neq l_2(o_j)$, for $1 \leq i, j \leq n$, $i \neq j$. Then for all such o, there is an $s \in S$ that can read o iff $n \leq |S|$.

- If a COI has n CDs, you need at least n subjects to access every object
- Proof sketch: If s can read o, it cannot read any o' in another CD in that COI (Axiom 8-2). As there are n such CDs, there must be at least n subjects to meet the conditions of the theorem.



Sanitized Data

- $v(o)$: sanitized version of object o
  - For purposes of analysis, place them all in a special CD in a COI containing no other CDs
- Axiom 8-5. $l_1(o) = \mathrm{l}_1(v(o))$ iff $l_2(o) = l_2(v(o))$

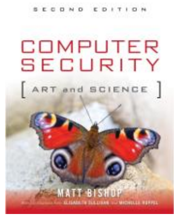


Which Objects Can Be Written?

Axiom 8-6. $s \in S$ can write to $o \in O$ iff the following hold simultaneously

1. $H(s, o)$
2. There is no $o' \in O$ with $H(s, o')$, $l_2(o) \neq l_2(o')$, $l_2(o) \neq l_2(v(o))$, $l_2(o') = l_2(v(o))$.

- Allow writing iff information cannot leak from one subject to another through a mailbox
- Note handling for sanitized objects



How Information Flows

Definition: information may flow from *o* to *o′* if there is a subject such that $H(s, o)$ and $H(s, o')$.

- Intuition: if *s* can read 2 objects, it can act on that knowledge; so information flows between the objects through the nexus of the subject
- Write the information flow between o and *o′* as (o, o')

Key Result

Theorem 8-3: Set of all information flows is

$$\{ (o, o') \mid o \in O \land o' \in O \land l_2(o) = l_2(o') \lor l_2(o) = l_2(v(o)) \}$$

Sketch of proof: Definition gives set of flows:

$$F = \{(o, o') \mid o \in O \land o' \in O \land \exists\, s \in S \text{ such that } H(s, o) \land H(s, o'))\}$$

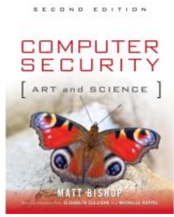
Axiom 8-6 excludes the following flows:

$$X = \{ (o, o') \mid o \in O \land o' \in O \land l_2(o) \neq l_2(o') \land l_2(o) \neq l_2(v(o)) \}$$

So, letting F^* be transitive closure of F,

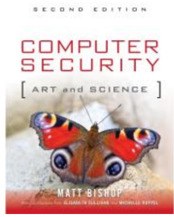
$$F^* - X = \{(o, o') \mid o \in O \land o' \in O \land \neg(l_2(o) \neq l_2(o') \land l_2(o) \neq l_2(v(o))) \}$$

which is equivalent to the claim.



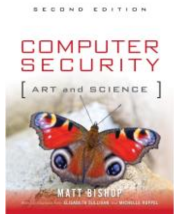
Aggressive Chinese Wall Model

- Assumption of Chinese Wall Model: COI classes are actually related to business, and those are partitions
  - Continuing bank and oil company example, the latter may invest in some companies, placing them in competition with banks
  - One bank may only handle savings, and another a brokerage house, so they are not in competition
- More formally: Chinese Wall model assumes the elements of O can be partitioned into COIs, and thence into CDs
  - Define *CIR* to be the conflict of interest relation induced by a COI
  - For $o, o' \in O$, if o, o' are in the same COI, then $(o, o') \in CIR$



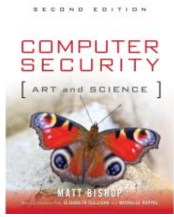
The Problem

- Not true in practice!
  - That is, in practice *CIR* does not partition the objects, and so not an equivalence class
  - Example: a company is not in conflict with itself, so $(o, o) \notin CIR$
  - Example: company *c* has its own private savings unit; *b* bank that does both savings and investments; oil company *g* does investments. So $(c, b) \in CIR$ and $(b, g) \in CIR$, but clearly $(c, g) \notin CIR$



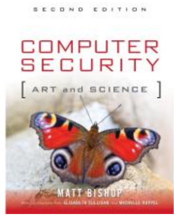
The Solution

- Generalize *CIR* to define COIs not based on business classes, so *GCIR* is the reflexive, transitive closure of *CIR*
- To create it:
  - For all $o \in O$, add (o, o) to *CIR*
  - Take the transitive closure of this
- Then (o, o') ∈ *GICR* iff there is an indirect information flow path between o and o'
  - Recall (o, o') ∈ *CIR* iff there is a direct information flow path between o, o'
- Now replace the COIs induced by *CIR* with generalized COIs induced by *GCIR*



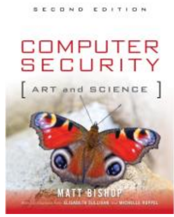
Compare to Bell-LaPadula

- Fundamentally different
  - CW has no security labels, Bell-LaPadula does
  - CW has notion of past accesses, Bell-LaPadula does not
- Bell-LaPadula can capture state at any time
  - Each (COI, CD) pair gets security category
  - Two clearances, *S* (sanitized) and *U* (unsanitized)
    - *S dom U*
  - Subjects assigned clearance for compartments without multiple categories corresponding to CDs in same COI class



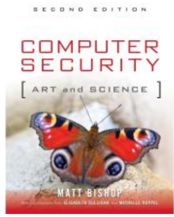
Compare to Bell-LaPadula

- Bell-LaPadula cannot track changes over time
  - Susan becomes ill, Anna needs to take over
    - C-W history lets Anna know if she can
    - No way for Bell-LaPadula to capture this
- Access constraints change over time
  - Initially, subjects in C-W can read any object
  - Bell-LaPadula constrains set of objects that a subject can access
    - Can’t clear all subjects for all categories, because this violates CW-simple security condition



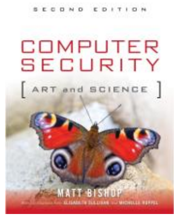
Compare to Clark-Wilson

- Clark-Wilson Model covers integrity, so consider only access control aspects
- If “subjects” and “processes” are interchangeable, a single person could use multiple processes to violate CW-simple security condition
  - Would still comply with Clark-Wilson Model
- If “subject” is a specific person and includes all processes the subject executes, then consistent with Clark-Wilson Model



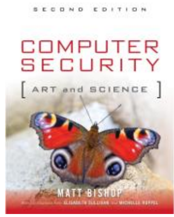
Clinical Information Systems Security Policy

- Intended for medical records
  - Conflict of interest not critical problem
  - Patient confidentiality, authentication of records and annotators, and integrity are
- Entities:
  - Patient: subject of medical records (or agent)
  - Personal health information: data about patient’s health or treatment enabling identification of patient
  - Clinician: health-care professional with access to personal health information while doing job



Assumptions and Principles

- Assumes health information involves 1 person at a time
    - Not always true; OB/GYN involves father as well as mother
- Principles derived from medical ethics of various societies, and from practicing clinicians

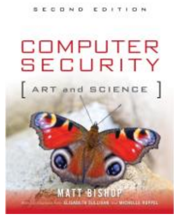


Access

- Principle 1: Each medical record has an access control list naming the individuals or groups who may read and append information to the record. The system must restrict access to those identified on the access control list.
  - Idea is that clinicians need access, but no-one else. Auditors get access to copies, so they cannot alter records

Access

- Principle 2: One of the clinicians on the access control list must have the right to add other clinicians to the access control list.
  - Called the *responsible clinician*



Access

- Principle 3: The responsible clinician must notify the patient of the names on the access control list whenever the patient’s medical record is opened. Except for situations given in statutes, or in cases of emergency, the responsible clinician must obtain the patient’s consent.
  - Patient must consent to all treatment, and must know of violations of security

Access

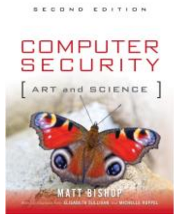
- Principle 4: The name of the clinician, the date, and the time of the access of a medical record must be recorded. Similar information must be kept for deletions.
  - This is for auditing. Don’t delete information; update it (last part is for deletion of records after death, for example, or deletion of information when required by statute). Record information about all accesses.

Creation

- Principle: A clinician may open a record, with the clinician and the patient on the access control list. If a record is opened as a result of a referral, the referring clinician may also be on the access control list.
  - Creating clinician needs access, and patient should get it. If created from a referral, referring clinician needs access to get results of referral.

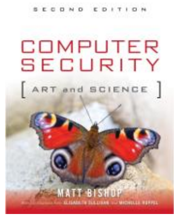
Deletion

- Principle: Clinical information cannot be deleted from a medical record until the appropriate time has passed.
  - This varies with circumstances.



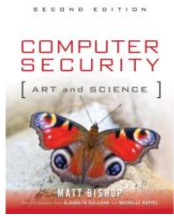
Confinement

- Principle: Information from one medical record may be appended to a different medical record if and only if the access control list of the second record is a subset of the access control list of the first.
  - This keeps information from leaking to unauthorized users. All users have to be on the access control list.



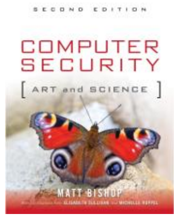
Aggregation

- Principle: Measures for preventing aggregation of patient data must be effective. In particular, a patient must be notified if anyone is to be added to the access control list for the patient's record and if that person has access to a large number of medical records.
    - Fear here is that a corrupt investigator may obtain access to a large number of records, correlate them, and discover private information about individuals which can then be used for nefarious purposes (such as blackmail)



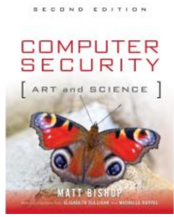
Enforcement

- Principle: Any computer system that handles medical records must have a subsystem that enforces the preceding principles. The effectiveness of this enforcement must be subject to evaluation by independent auditors.
  - This policy has to be enforced, and the enforcement mechanisms must be auditable (and audited)



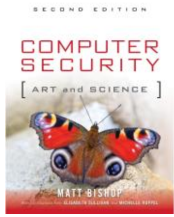
Compare to Bell-LaPadula

- Confinement Principle imposes lattice structure on entities in model
  - Similar to Bell-LaPadula
- CISS focuses on objects being accessed; Bell-LaPadula on the subjects accessing the objects
  - May matter when looking for insiders in the medical environment



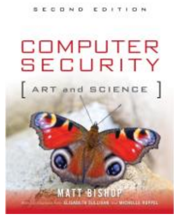
Compare to Clark-Wilson

- CDIs are medical records
- TPs are functions updating records, access control lists
- IVPs certify:
  - A person identified as a clinician is a clinician;
  - A clinician validates, or has validated, information in the medical record;
  - When someone is to be notified of an event, such notification occurs; and
  - When someone must give consent, the operation cannot proceed until the consent is obtained
- Auditing (CR4) requirement: make all records append-only, notify patient when access control list changed



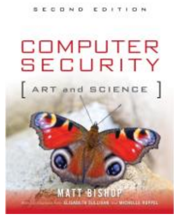
Originator Controlled Access Control

- Problem: organization creating document wants to control its dissemination
  - Example: Secretary of Agriculture writes a memo for distribution to her immediate subordinates, and she must give permission for it to be disseminated further. This is “originator controlled” (here, the “originator” is a person).



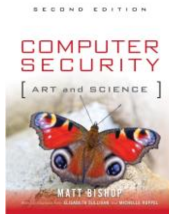
Requirements

- Subject $s \in S$ marks object $o \in O$ as ORCON on behalf of organization X. X allows o to be disclosed to subjects acting on behalf of organization Y with the following restrictions:
    1. o cannot be released to subjects acting on behalf of other organizations without X's permission; and
    2. Any copies of o must have the same restrictions placed on it.



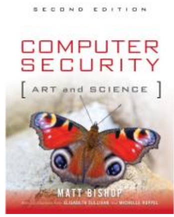
DAC Fails

- Owner can set any desired permissions
  - This makes 2 unenforceable



MAC Fails

- First problem: category explosion
  - Category C contains o, X, Y, and nothing else. If a subject $y \in Y$ wants to read o, $x \in X$ makes a copy o'. Note o' has category C. If y wants to give $z \in Z$ a copy, z must be in Y—by definition, it's not. If x wants to let $w \in W$ see the document, need a new category C' containing o, X, W.
- Second problem: abstraction
  - MAC classification, categories centrally controlled, and access controlled by a centralized policy
  - ORCON controlled locally

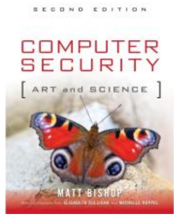


Combine Them

- The owner of an object cannot change the access controls of the object.
- When an object is copied, the access control restrictions of that source are copied and bound to the target of the copy.
  - These are MAC (owner can’t control them)
- The creator (originator) can alter the access control restrictions on a per-subject and per-object basis.
  - This is DAC (owner can control it)

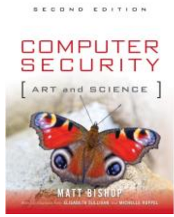
Digital Rights Management (DRM)

- The persistent control of digital content
- Several elements:
  - Content: information being protected
  - License: token describing the uses allowed for the content
  - Grant: part of a license giving specific authorizations to one or more entities, and (possibly) conditions constraining the use of the grant
  - Issuer: entity issuing the license
  - Principal: identification of an entity, used in a license to identify to whom the license applies
  - Device: mechanism used to view the content



Example: Movie Distribution by Downloading

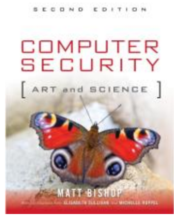
- Content: movie itself
- License: token binding palying the movie to the specific downloaded copy
- Grant: movie can be played on some specific set of equipment provided the equipment is located in a geographical area
- Issuer: movie studio
- Principal: user who downloaded the movie
- Device: set of equipment used to play the movie; it manages the licenses, principle, and any copies of the movie



Relationships

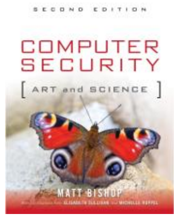
Elements related, and the relationship must satisfy all of:

1. The system must implement controls on the use of the content, constraining what users can do with the content
   - Encrypting the content and providing keys to authorized viewers fails this, as the users can distribute the keys indiscriminently
2. The rules that constrain the users of the content must be associated with the content, not the users
3. The controls and rules must persist throughout the life of the content, regardless of how it is distributed and to whom it is distributed



Conditions

- Stated using a rights expression language
- Example: Microsoft's ReadyPlay uses a language supporting temporal constraints such as
  - Allowing the content to be viewed over a specific period of time
  - Allowing a validity period for the license
  - Allowing constraints on copying, transferring, converting the content
  - Allowing geographical constraints
  - Allowing availability constraints (for example, content can't be played when being broadcast)



Example: Microsoft PlayReady DRM

Setup

- Content is enciphered using AES
- Key made available to a license server, encrypted content to a distribution server

Play

- Client downloads content, requests license
- License server authenticates client; on success, constructs license and sends it
- Client checks the constraints and, if playback allowed, uses the key in the license to decipher content

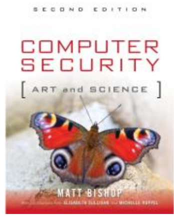
Example: Apple's FairPlay DRM

Set up system to play using iTunes

- iTunes generates globally unique number, sends it to Apple's servers
- Servers add it to list of systems authorized to play music for that user
  - At most 5 systems at a time can be authorized

Obtain content using iTunes

- Content enciphers by AES with a master key
- Master key locked with a randomly generated user key from iTunes
- iTunes sends user key to Apple server; stored there and in iTunes, encrypted



Example: Apple's FairPlay DRM

Play content using iTunes

- iTunes decrypts user key
- iTunes uses user key to decrypt master key
- iTunes uses master key to decrypt content
- Note it need not contact Apple servers for authorization

Authorize new system

- Apple server sends that system all user keys stored on server

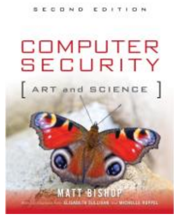
Example: Apple’s FairPlay DRM

Deauthorize system

- System deletes all locally stored user keys
- Notifies Apple servers to delete globally unique number from list of authorized computers

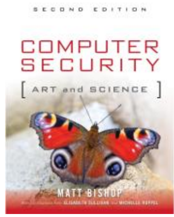
Copying content to another system

- Cannot be decrypted without user key, which is not copied



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.



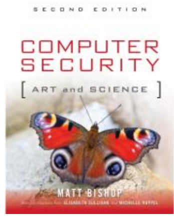
Definitions

- 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

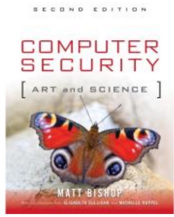
- Trainer can do all transactions that trainee can do (and then some). This means role r contains role r' ($r > r'$). So:

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

Separation of Duty

- 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:

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



RBAC Hierarchy

- $RBAC_0$: basic model (you just saw it)
- $RBAC_1$: adds role hierarchies to $RBAC_0$
- $RBAC_2$: adds constraints to $RBAC_0$
- $RBAC_3$: adds both role hierarchies, constraints to $RBAC_0$
    - It combines $RBAC_1$ and $RBAC_2$

RBAC$_0$, Formally

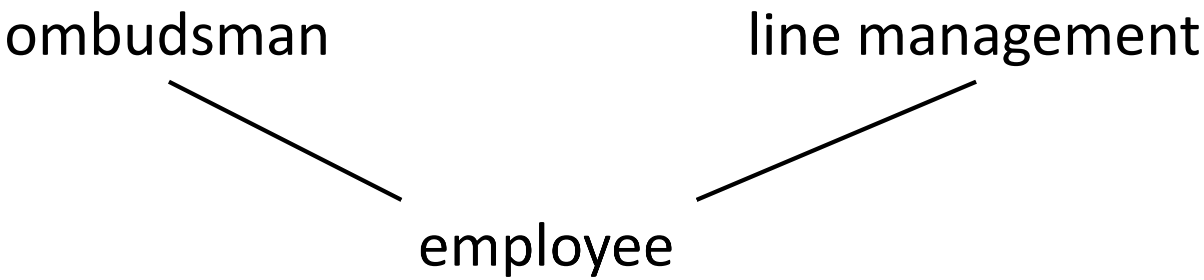
- Set of users U, roles R, permissions P, sessions S
- Relation $PA \subseteq P \times R$ mapping permissions to roles
- Relation $UA \subseteq U \times R$ mapping users to roles
- Function *user*: $S \rightarrow U$ mapping each session to a user
- Function *roles*: $S \rightarrow 2^R$ mapping each session $s \in S$ to a set of roles $roles(s) \subseteq \{ r \in R \mid (user(s), r) \in UA \}$, where s has permissions

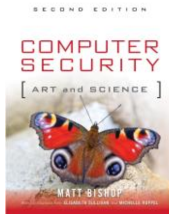
$$\bigcup_{r \in roles(s)} \{ p \in P \mid (p, r) \in PA \}$$

  - When a user assumes role r during session, r and hence the user assuming r gets the set of permissions associated with r

RBAC$_1$, Intuitively

- Add containment of roles to RBAC$_0$ (this is the hierarchy)
  - It's a partial ordering
- Each role less powerful than its containing role
  - Containing role contains job functions (permissions) of the contained role
- Can define *private roles* in which one role is subordinate to two others, and those two are not related



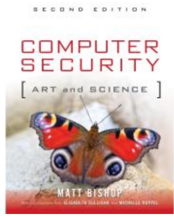


RBAC_1, Formally

- Set of users U, roles R, permissions P, sessions S
- Partial order $RH \subseteq R \times R$
  - Write $(r_1, r_2) \in R$ as $r_1 \geq r_2$
- Relation $PA \subseteq P \times R$ mapping permissions to roles
- Relation $UA \subseteq U \times R$ mapping users to roles
- Function *user*: $S \rightarrow U$ mapping each session to a user
- Function *roles*: $S \rightarrow 2^R$ mapping each session s $\in S$ to a set of roles $roles(s) \subseteq \{\, \mathrm{r} \in \mathrm{R} \mid (\exists r' \geq r)(user(s), r') \in UA\}$, where s has permissions

$$\bigcup_{r \in roles(s)} \{\, p \in P \mid (\exists r'' \geq r)(\, p, r'') \in PA\}$$

  - When a user assumes role *r* with subordinate role *r'* during session, *r* and hence the user assuming *r* gets the set of permissions associated with *r*, and hence with *r'*

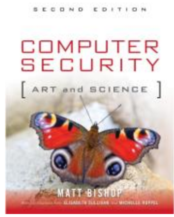


$RBAC_2$ and $RBAC_3$

- $RBAC_2$ adds constraints on values that components can assume to $RBAC_0$
  - Example: user can be in only one role at a time
  - Example: make 2 roles mutually exclusive
- RBAC3 provides both role hierarchies and constraints that determine allowable values for relations and functions
  - Combines $RBAC_1$ and $RBAC_2$
- Can be extended to manage role and privilege assignments
  - A set of administrative roles *AR* and permissions *AP* defined disjointly from *R* and *P*
  - Constraints allow $ap \in AP$ to be assigned to $ar \in AR$ only, and $p \in P$ to $r \in R$ only

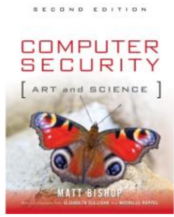
Role Engineering

- *Role engineering*: defining roles and determining needed permissions
- Often used when two organizations using RBAC merge
  - Roles in one organization rarely overlap with roles in other
  - Job functions often do overlap
- *Role mining*: analyzing existing roles, permission assignments to determine optimal assignment of permissions to roles
  - *NP*-complete, but in practice optimal solutions can be approximated or produced



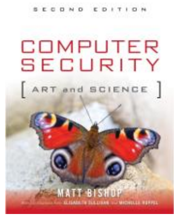
Break-the-Glass Policies

- Motivation: when security requirements conflict, some access controls may need to be overwritten in an unpredictable manner
    - Example: a doctor may need access to a medical record to treat someone, yet that person is unable to give consent (without which access would be denied)
- User overrides the denial
    - Controls notify some people about the override
    - Controls log override for later audit



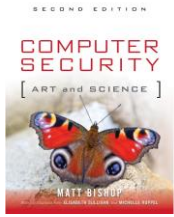
Example: Rumpole

- Implements a break-the-glass policy
- *Evidential rules*: how to assemble evidence to create context for request
- *Break-glass rules*: define permissions
    - Includes constraints such as obligations to justify need for actions
- *Grant policies*: how rules are combined to determine whether to grant override



Example: Rumpole Enforcement Model

- *Request*: subject, desired action, resource, obligations acceptable to subject
- Decision point:
  - Grants request
  - Denies request
  - Returns request with set of obligations subject must accept; subject then can send a new request with that set of obligations, if they are acceptable



Key Points

- Hybrid policies deal with both confidentiality and integrity
  - Different combinations of these
- ORCON model neither MAC nor DAC
  - Actually, a combination
- RBAC model controls access based on functionality
- Break-the-glass model handles exceptional circumstances that the access control model does not account for