PURDUE UNIVERSITY

CS52600: Information Security

Policy

13 September 2010

Prof. Chris Clifton

Security Policy

- What is a security policy?
  - Defines what it means for a system to be secure
- Formally: Partition system into
  - Secure (authorized) states
  - Non-secure (unauthorized) states
- Secure system:
  - Starts in authorized state
  - **Can't** enter unauthorized state

2

Secure System - Example

- Is this Finite State Machine Secure?
  - *A* and *B* are authorized states
  - *B* is start state
  - *C* is start state
  - *A*, *B*, and *C* are authorized states
- A PC is just a big Finite State Machine:
  - *Is Microsoft Windows Secure?*

3

Additional Definitions:

- Breach of security
  - Transition causing system to enter unauthorized state
- Let *X* be a set of entities, *I* be information.
  - *I* has **confidentiality** with respect to *X* if no member of *X* can obtain information on *I*
  - *I* has **integrity** with respect to *X* if all members of *X* trust *I*
  - *I* has **availability** with respect to *X* if all members of *X* can access *I*
- Security Policy defines all of the above
  - Now just need to define obtain, trust, access

4

Confidentiality Policy

- What does “obtain” information mean?
- Formally: *information flow*
  - Transfer of rights
  - Transfer of information without transfer of rights
- Model often depends on trust
  - Parts of system where information *could* flow
  - Trusted entity must participate to enable flow
- Highly developed in Military/Government

5

Integrity Policy

- Defines how information can be altered
  - Entities allowed to alter data
  - Conditions under which data can be altered
  - Limits to change of data
- Examples:
  - Purchase over $1000 requires signature
  - Check over $10,000 must be signed by two officers
    - *Separation of duties*
- Highly developed in commercial world

6

Availability Policy

- Defines what it means for information to be accessible
  - Time limits (quality of service)
  - Access methods
    - On-line access vs. telephone vs. mail
- Integrity and availability may interrelate
  - Fast old copy vs. slow current version

7

Security Mechanism

- Policy describes what is allowed
- Mechanism enforces (part of) policy

The two need not be the same!

- Example Policy: Students should not copy homework
  - Mechanism: Disallow access to files owned by other users
- Does mechanism enforce policy?
  - Is mechanism too strict?

8

Security Model

- Security Policy: What is/isn't authorized
- Problem: Policy specification often informal
  - Implicit vs. Explicit
  - Ambiguity
- Security Model: Model that represents a particular policy (policies)
  - Model must be explicit, unambiguous

9

Trust

- Trusted Entity
  - Entity that can violate security
- What are typical Trusted Entities?
  - People with access to information
  - System developers
  - Hardware
  - ?

Where does it end?

10

Common Mechanisms: Access Control

- Discretionary Access Control (DAC)
  - Owner determines access rights
  - Typically *identity-based access control*: Owner specifies other users who have access
- Mandatory Access Control (MAC)
  - Rules specify granting of access
  - Also called *rule-based access control*
- Originator Controlled Access Control (ORCON)
  - Originator controls access
  - *Originator need not be owner!*
- Role Based Access Control (RBAC)
  - Identity governed by role user assumes

11

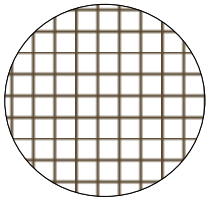
Policy Languages

- Security Policy isn't enough
  - Need to express it to get Policy Model
- Policy Language: Means of expression
- High-level: Independent of mechanisms
  - Example: Domain-Type Enforcement Language
    - Subjects partitioned into domains
    - Objects partitioned into types
    - Each domain has set of rights over each type
- Low-level: Acts on mechanisms
  - Example: Tripwire: Flags what has changed
    - Configuration file specifies settings to be checked
    - History file keeps old (good) example

12

Creating a Secure System

- Can we make it secure?
  - *Easy!*
- But can we make it precise?
- Next Time: Model allowing us to capture this



precise

set of secure states

set of reachable states

13

PURDUE UNIVERSITY

CS52600: Information Security

Policy

15 September 2010

Prof. Chris Clifton

Modeling Secure/Precise: Confidentiality (Jones and Lipton '75)

- What are we modeling? A program
  - p: $I_1 \times \ldots \times I_n \rightarrow R$ is a program
    - Defined in terms of inputs and outputs
  - Goal: Determine if p can violate confidentiality
- Observability
  - Output of function $p(i_1,\ldots,i_n)$ encodes all *available* information on inputs $i_1,\ldots,i_n$
  - Output may include things not normally thought of as part of function result
    - Data accessed
    - Timing
    - *Anything that can be observed*

15

Modeling Secure/Precise: Confidentiality

- Protection Mechanism
  m: $I_1 \times \ldots \times I_n \rightarrow R \times E$ such that:
  - $m(i_1,\ldots,i_n) = p(i_1,\ldots,i_n)$, or
    - Acceptable result
  - $m(i_1,\ldots,i_n) \in E$
    - Protection violation (result of p would disclose confidential information)
- Confidentiality Policy for program p:
  c: $I_1 \times \ldots \times I_n \rightarrow A$
  - $A \subseteq I_1 \times \ldots \times I_n$ is inputs that can be revealed

16

Modeling **Secure**/Precise: Confidentiality

- Secure Program: Given confidentiality policy *c* for program *p*, and mechanism *m* for *p*
  - *m* is secure iff $\exists$ *m'*:$A \rightarrow R \times E$ such that $\forall\ i_k \in I_k$, $m(i_1, \ldots, i_n) = m'(c(i_1, \ldots, i_n))$
- What does this mean?
  - Must be able to generate results from non-confidential inputs

17

Modeling Secure/**Precise**: Confidentiality

- m_1 as precise as m_2 if $\forall\ i_k \in I_k$
  - $m_2(i_1, \ldots, i_n) = p(i_1, \ldots, i_n) \Rightarrow$ $m_1(i_1, \ldots, i_n) = p(i_1, \ldots, i_n)$
  - Write $m_1 \approx m_2$
- m_1 more precise than m_2 if $\exists\ i_k \in I_k$ s.t.
  - $m_1(i_1, \ldots, i_n) = p(i_1, \ldots, i_n)$
  - $m_2(i_1, \ldots, i_n) \neq p(i_1, \ldots, i_n)$
  - Write $m_1 \sim m_2$

18

Modeling Secure/Precise: Confidentiality

- $m_3 = m_1 \cup m_2$ defined as
  - $p(i_1, \ldots, i_n)$ when $m_1(i_1, \ldots, i_n) = p(i_1, \ldots, i_n)$ or $m_2(i_1, \ldots, i_n) = p(i_1, \ldots, i_n)$
  - else $m_1(i_1, \ldots, i_n)$
  - *Less restrictive than either*
- Theorem: if m_1 and m_2 secure,
  - $m_1 \cup m_2$ secure
  - $m_1 \cup m_2 \approx m_1$ and $m_1 \cup m_2 \approx m_2$

19

Modeling Secure/Precise: Confidentiality

- Theorem: if m_1 and m_2 secure,
  - $m_1 \cup m_2$ secure
  - $m_1 \cup m_2 \approx m_1$ and $m_1 \cup m_2 \approx m_2$
- Proof Sketch
  - Result = result of p
    - Only if same as m_1 or m_2
    - m_1 and m_2 secure
  - Result = result of m_1
    - m_1 secure

20

Modeling Secure/Precise: Confidentiality

- Theorem: Given *p* and *c*, ∃ a precise, secure mechanism *m** such that ∀ secure *m* for *p* and *c*, *m** ≈ *m*
  - Proof: Induction from previous theorem
- Theorem: Impossible to construct *m**
  - Proof: Reduction from Halting Problem
  - *c* = constant function (reveal no information)
  - *p* such that *m* either non-constant or undefined
    - Non-constant not allowed
    - Undefined corresponds to *p* halts
  - Contradiction: Either *m* non-constant, or we know *p* halts
    - *p* defined as in halting problem

21

Modeling Secure/Precise: Integrity

- Integrity Policy: Set of valid outputs for given input
- Mechanism: Given program, policy, produce output allowed by policy
  - Program output if allowed
  - Valid output otherwise
    - probably includes error
- Precision: Does mechanism produce program result whenever allowed?

22

PURDUE UNIVERSITY

CS52600: Information Security

Bell-LaPadula Model

17 September, 2010

Prof. Chris Clifton

Confidentiality Policy: Bell-LaPadula Model

- Formally models military-style classification
  - Multi-level access control
- Mandatory Access Control
  - Clearance
- Discretionary Access Control
  - Need to Know
- First real attempt to model and prove security of real systems

24

Bell-LaPadula: Basics

- Mandatory access control (Security Level)
  - Subject has clearance $L(S) = l_s$
  - Object has classification $L(O) = l_o$
  - Clearance/Classification ordered
    - $l_i < l_{i+1}$
- Discretionary access control
  - Matrix: Subject has read (write) on Object
- Need both to perform operation

25

Access Rules (Preliminary)

- *S* can read *O* if and only if
  - $l_o \leq l_s$ and
  - *S* has discretionary read access to *O*
- *S* can write *O* if and only if
  - $l_s \leq l_o$ and
  - *S* has discretionary write access to *O*
- Secure system: One with above properties
- Theorem: Let Σ be a system with secure initial state σ_0, *T* be a set of state transformations
  - If every element of *T* follows rules, every state σ_i secure
  - Proof - induction

26

Categories

- Total order of classifications not flexible enough
  - Alice cleared for missiles
  - Bob cleared for warheads
  - Both cleared for targets
- Solution: Categories
  - Set of compartments
  - *S* can access *O* if $C(S) \supseteq C(O)$
- Combining with clearance:
  - (*L*,*C*) *dominates* (*L'*,*C'*) $\Leftrightarrow$ *L'* ≤ *L* and $C' \subseteq C$
  - Induces lattice instead of levels

27

Access Rules

- Simple Security Condition: *S* can read *O* if and only if
  - *S dom O* and
  - *S* has discretionary read access to *O*
- *-Property: *S* can write *O* if and only if
  - *O dom S* and
  - *S* has discretionary write access to *O*
- Secure system: One with above properties
- Theorem: Let Σ be a system with secure initial state σ_0, *T* be a set of state transformations
  - If every element of *T* follows rules, every state σ_i secure

28

Problem: No write-down

Cleared subject can't talk to non-cleared subject

- Any write from l_i to l_k, $i > k$, would violate *-property
  - Subject at l_i can only write to l_i and above
- Any read from l_k to l_i, $i > k$, would violate simple security property
  - Subject at l_k can only read from l_k and below
- Subject at level i can't write something readable by subject at k
  - Not very practical

29

Solution: Active Level

- Subject has maximum and current level
  - Maximum must dominate current
  - Rules apply to current security level
- What does this mean for security?
  - Subject is trusted when maximum ≠ current

30

Instantiating the Model

- What must be defined?
  - Levels
  - Categories
  - Subjects, objects
  - *Meaning of read and write*

31

Formalizing Bell-LaPadula: System consists of

- Set of subjects S
- Set of objects O
- Set of rights P = { **r** (read), **a** (write), **w** (read/write), **e** (empty)}
- Set of possible access control matrices M
- Set of classifications C
- Set of categories K
- Levels $L = C \times K$
- Set F of tuples (f_s, f_o, f_c) representing
  - f_s: Maximum security level of subject
  - f_o: Security level of object
  - f_c: Current security level of subject

33

Formalizing Bell-LaPadula

- Objects in a hierarchy h: $O \rightarrow P(O)$
  - $o_i \neq o_j \Rightarrow h(o_i) \cap h(o_j) = \varnothing$ (no two nodes at same point)
  - There is no $\{ o_1, o_2, \ldots, o_k \} \subseteq O$ such that $\forall i = 1,\ldots,k$, $o_{i+1} \in h(o_i)$ and $o_{k+1} = o_1$ (no cycles)
- State $v \in V$ is a 4-tuple (b,m,f,h)
  - $b \in P(S \times O \times P)$ indicates which subjects can access which objects and what the rights are
- R denotes requests for access
- D set of outcomes
  - **y**es, **n**o, **i**llegal, err**o**r
- Actions $W \subseteq R \times D \times V \times V$
  - Request leads to outcome, moving from one state to another
- System $\Sigma(R, D, W, z_0) \subseteq R^N \times D^N \times V^N$
  - Set of states that result from a given set of actions
  - $(r,d,v,v') \in W$ an action of Σ iff $\exists$ time t, $(x,y,z) \in \Sigma$ such that $(r,d,v,v') = (x_t,y_t,z_t,z_{t-1})$

34

A System is Secure if it Satisfies:

- Simple security condition satisfied for $(s, o, p) \in S \times O \times P$ relative to f iff
  - p = **e** or p = **a**
  - p = **r** or p = **w** and $f_s(s)$ *dom* $f_o(o)$
- *-property satisfied for (b, m, f, h) iff $\forall\ s \in S$
  - $b(s{:}\mathbf{a}) \neq \varnothing \Rightarrow [\forall\ o \in b(s{:}\ \mathbf{a})\ [f_o(o)\ dom\ f_c(s)]]$
  - $b(s{:}\mathbf{w}) \neq \varnothing \Rightarrow [\forall\ o \in b(s{:}\ \mathbf{w})\ [f_o(o) = f_c(s)]]$
  - $b(s{:}\mathbf{r}) \neq \varnothing \Rightarrow [\forall\ o \in b(s{:}\ \mathbf{r})\ [f_c(s)\ dom\ f_o(o)]]$
- Discretionary security property satisfied for (b, m, f, h) iff $\forall\ (s, o, p) \in b, p \in m[s,o]$

35

Theorem: Simple Security Condition

- $\Sigma(R, D, W, z_0)$ satisfies simple security condition for any secure z_0 iff $\forall$actions $(r, d, (b, m, f, h), (b', m', f', h'))$, W satisfies
  - $\forall (s, o, p) \in b - b'$ ssc satisfied relative to f
  - $\forall (s, o, p) \in b'$ not satisfying ssc relative to f,
    - $(s, o, p) \notin b$
- Proof:
  - If: Induction – each transition maintains secure state
  - Only if: A transition from a secure to non-secure state requires an action not satisfying ssc

Theorem: *-property

- $\Sigma(R, D, W, z_0)$ satisfies *-property relative to $S' \subseteq S$ for any secure z_0 iff $\forall$actions $(r, d, (b, m, f, h), (b', m', f', h'))$, W satisfies $\forall s \in S'$
  - $\forall (s, o, p) \in b - b'$, *-property satisfied for S'
  - $\forall (s, o, p) \in b$ not satisfying *-property for S',
    - $(s, o, p) \notin b'$
- Proof similar
- Similar theorem for ds-property
- Basic Security Theorem: $\Sigma(R, D, W, z_0)$ secure if z_0 secure and W satisfies conditions of above theorems

PURDUE UNIVERSITY

CS526: Information Security
Chris Clifton

20 September 2010

Bell-LaPadula Model:

Application to Multics

Transformation Rules

- $\rho: R \times V \rightarrow D \times V$
  - Takes request and state, produces outcome and state
- ρ is ssc-preserving if $\forall (r,v) \in R \times V$ where v satisfies ssc rel f,
  - $\rho(r,v) = (d,v') \Rightarrow v'$ satisfies ssc rel f'
- $\omega = \{\rho_1, \ldots, \rho_m\}$. For $r \in R$, $d \in D$, and $v, v' \in V$
  - $(r, d, v, v') \in W(\omega) \Leftrightarrow d \neq \mathbf{i}$ and $\exists 1 \leq i \leq m$ such that $\rho_i(r,v) = (d,v')$
  - If request is legal and only one rule causes change, corresponding action exists

40

Transformation Theorems

- Let ω be a set of ssc-preserving rules, z_0 a state satisfying ssc.
  - $\Sigma(R, D, W(\omega), z_0)$ satisfies the simple security condition
- Proof: Assume z_t not secure, z_{t-1} secure.
  - Forces rule not meeting ssc-preserving rule definition
- Similar definitions, theorems for other properties

41

Multics: Background

- Early time-sharing operating system
  - Previous computers processed batch jobs one at a time
    - Security policy easy to enforce
  - Time sharing introduces new challenges
- Five categories of rules
  - Requests for access
  - Permission granting
  - Object reclassification
  - Delete object
  - Subject reclassification
- Trusted users: *-property not enforced

43

Rule List (all rules return *yes*/*no*)

- Access requests: $R^{(1)} = Q \times S \times O \times M$
  - get-read, get-append, get-execute, get-write
  - release-read, append, execute, write
- Permission granting: $R^{(2)} = S \times Q \times S \times O \times M$
  - give/rescind read/append/execute/write
- Object reclassification: $R^{(3)} = Q \times S \times O \times L$
  - create-object, change-object-security-level
- Delete object: $R^{(4)} = S \times O$
  - delete-object-group deletes object and children
- Subject reclassification: $R^{(5)} = S \times L$
  - change-subject-current-security-level

44

Modeling with Bell-LaPadula: *get-read*

- $r = (get, s, o, \mathbf{r}) \in R^{(1)}$ *request*
- $v = (b, m, f, h)$ *system state*
- **if** ($r \notin \Delta(\rho_1)$) **then** $\rho_1(r,v) = (\mathbf{i}, v)$ *bad arguments*
  else if ($f_s(s)$ *dom* $f_o(o)$ *ssc preserving*
  and [$s \in S_T$ **or** $f_c(s)$ *dom* $f_o(o)$] **-property*
  and $\mathbf{r} \in m[s,o]$) *discretionary access control*
  then $\rho_1(r,v) = (\mathbf{y}, (b \cup \{ (s, o, \mathbf{r}) \}, m, f, h))$
  else $\rho_1(r,v) = (\mathbf{n}, v)$
- Theorem: *get-read* is secure
  - Assume *v* secure
  - Either *v'* = *v*, or *v'* = *v* with { (*s*, *o*, **r**) } added to accesses
    - (*s*, *o*, **r**) must satisfy security properties to reach where it is added
- Similar rules for *get-append*, *execute*, *write*

45

Modeling with Bell-LaPadula: *give-read*

- Idea: *give-read* allowed if requester can write to parent of object
  - Based on hierarchy of objects
  - Special handling of root: set of subjects allowed requests
- $r = (s_1, give, s_2, o, \mathbf{r}) \in R^{(2)}$, $v = (b, m, f, h)$
- **if** (r $\notin \Delta(\rho_r)$) **then** $\rho_6(r,v) = (\mathbf{i}, v)$
  else if ([$o \neq root(o)$ **and** $parent(o) \neq root(o)$ **and** $parent(o) \in b(s_1: \mathbf{w})$]
  or [$parent(o) = root(o)$ **and** $canallow(s_1, o, v)$]
  or [$o = root(o)$ **and** $canallow(s_1, root(o), v)$])
  then $\rho_6(r,v) = (\mathbf{y}, (b, m + m[s_2,o] \leftarrow \mathbf{r}, f, h))$
  else $\rho_6(r,v) = (\mathbf{n}, v)$
- Theorem: *give-read* is secure
  - Only possible change is to access control matrix
    - No effect on mandatory access control policies
  - *discretionary security property* definition: $\forall (s,o,p) \in b, p \in m[s,o]$
    - $b = b'$, $m \subseteq m'$

46

Multics: Other Rules

- State $v = (b, m, f, h)$
- Create/reclassify
  - What needs to be checked?
- Delete
  - Does this affect mandatory access control?
- Change security level
- How do we model these?
  - *See the homework*

47

Will this guarantee Multics is Secure?

- Demonstrates design secure
  - Implementation must enforce rules
- Reduces problem
  - Smaller "compartments" that must be validated
- Examples
  - Rule returns right answer
  - access only if in *b*
  - Others?

48

Announcements

- Mid-Semester Course Evaluation
  - Open to Friday 10/1, I get summary results Monday after that
  - https://portals.cs.purdue.edu , Course Evaluation, Mid-Semester Evaluations
- Midterm Wednesday 10/20
  - In class?
  - Or evening exam (in a larger room)?
- Assignment 4 should be out today
- Guest lecturer, Prof. Ninghui Li, Friday and Monday.

50

PURDUE UNIVERSITY

CS526: Information Security
Chris Clifton

22 September 2010

Bell-LaPadula Model:

Tranquility

Tranquility

- Classification changes make things difficult
  - Declassification violates properties
  - What about increasing classification of object?
- Principle of Strong Tranquility
  - Security levels do not change
- Principle of Weak Tranquility
  - Security level changes cannot violate policy

52

Follow-on work: McLean

- Problems with Bell-LaPadula
- Basically, Bell-LaPadula trivial
  - Definitions capture policy
  - Only thing interesting is showing induction
- McLean proposed very similar policy
  - Provably bad
  - But not easy to see why not okay by Bell-LaPadula
- Key: Axiomatic vs. "models real world" definitions of security
- *Read discussion*

53

Integrity Policy

- Principles:
  - Separation of Duty: Single person can't mess up the system
    - No coding on live system
  - Separation of function
    - No development on production data
  - Auditing
    - Controlled/audited process for updating code on production system
- This enables validated code to maintain integrity
  - *But how do we ensure we've accomplished these?*
  - *Is this overkill?*

54

Biba's Integrity Policy Model

- Based on Bell-LaPadula
  - Subject, Objects
  - Ordered set of Integrity Levels
    - Higher levels are more reliable/trustworthy
- Information transfer path:
  Sequence of subjects, objects where
  - s_i **r** o_i
  - s_i **w** o_{i+1}

56

Policies

- Ring Policy
  - s **r** o
  - s **w** $o \Leftrightarrow i(o) \leq i(s)$
  - s_1 **x** $s_2 \Leftrightarrow i(s_2) \leq i(s_1)$
- Low-Water-Mark Policy
  - s **r** $o \Rightarrow i'(s) = min(i(s), i(o))$
  - s **w** $o \Leftrightarrow i(o) \leq i(s)$
  - s_1 **x** $s_2 \Leftrightarrow i(s_2) \leq i(s_1)$
- Biba's Model: Strict Integrity Policy
  - s **r** $o \Leftrightarrow i(s) \leq i(o)$
  - s **w** $o \Leftrightarrow i(o) \leq i(s)$
  - s_1 **x** $s_2 \Leftrightarrow i(s_2) \leq i(s_1)$
- Theorem for induction similar to Bell-LaPadula

57

Lipner: Integrity Matrix

- Security Levels
  - Audit: AM
    - Audit/management functions
  - System Low: SL
    - Everything else
- Categories
  - **D**evelopment
  - **P**roduction **C**ode
  - **P**roduction **D**ata
  - **S**ystem **D**evelopment
  - Software **T**ools
    - Not related to sensitive/protected data
- Follow Bell-LaPadula security properties

58

Lipner: Integrity Matrix

- Users:
  - Ordinary (SL,{PC, PD})
  - Developers (SL,{D,T})
  - System Programmers (SL,{SD, T})
  - Managers (AM,{D,PC,PD,SD,T})
  - Controllers (SL,{D,PC,PD,SD,T}
- Objects
  - Development code/data (SL,{D,T})
  - Production code (SL,{PC})
  - Production data (SL,{PC,PD})
  - Tools (SL,{T})
  - System Programs (SL,∅)
  - System Program update (SL,{SD,T})
  - Logs (AM, {…})

59

Clark/Wilson

- Transaction based
  - State before/after transaction
- Consistency definitions
  - What states of system are acceptable
- Well-Formed Transaction
  - State before transaction consistent ⇒ state after transaction consistent
- Components
  - **C**onstrained **D**ata **I**tems
  - **U**nconstrained **D**ata **I**tems
  - **I**ntegrity **V**erification **P**rocedures
  - **T**ransformation **P**rocedures

60

Clark/Wilson: Certification Rules

- When any IVP is run, it must ensure all CDIs are in valid state
- A TP transforms a set of CDIs from a valid state to another valid state
  - Must have no effect on CDIs not in set
- Relations between (*user*, TP, {CDI}) must support separation of duty
- All TPs must log undo information to append-only CDI
- A TP taking a UDI as input must either reject it or transform it to a CDI

61

Clark/Wilson: Enforcement Rules

- System must maintain certified relations
  - TP/CDI sets enforced
- System must control users
  - *user*/TP/CDI mappings enforced
- Users must be authenticated to execute TP
- Only certifier of a TP may change associated CDI set

62

Domain-specific Policy Models

- Military Confidentiality
  - Bell-LaPadula
- Database Integrity
  - Clark/Wilson
- Corporate Anti-Trust
  - Chinese Wall
- Clinical Information Systems
- Others?

63

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

64

Chinese Wall Model

- Supports confidentiality and integrity
- Models conflict of interest
  - object sets *CD*
  - conflict of interest sets *COI*
- Principle: Information can't flow between items in a COI set
  - *S* can read *O* $\Leftrightarrow$ one of the following holds
    - $\exists O' \in$ ***P**reviousRead*(*S*) such that *CD*(*O'*) = *CD*(*O*)
    - $\forall O'$, $O' \in PR(S) \Rightarrow COI(O') \neq COI(O)$, or
    - *O* has been "sanitized"

See reading for more details

65

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

66

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

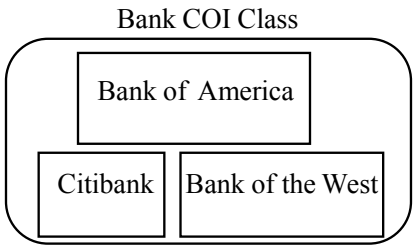
67

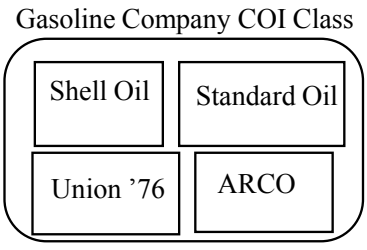
Definitions

- *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

68

Example





69

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

70

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′* ∈ *O*, *o′* ∈ *PR*(*s*) ⇒ *COI*(*o′*) ≠ *COI*(*o*)
     - Meaning *s* has not read any objects in *o*'s conflict of interest class
- Ignores sanitized data (see below)
- Initially, *PR*(*s*) = ∅, so initial read request granted

71

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

72

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

73

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

74

Formalism

- Goal: figure out how information flows around system
- *S* set of subjects, *O* set of objects, *L* = *C*×*D* set of labels
- I_1:*O*→*C* maps objects to their COI classes
- I_2:*O*→*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*

75

Axioms

- Axiom 7-1. For all $o, o' \in O$,
  if $I_2(o) = I_2(o')$, then $I_1(o) = I_1(o')$
  - CDs do not span COIs.
- Axiom 7-2. $s \in S$ can read $o \in O$ iff,
  for all $o' \in O$ such that $H(s, o')$, either
  $I_1(o') \neq I_1(o)$ or $I_2(o') = I_2(o)$
  - 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.

76

More Axioms

- Axiom 7-3. $\neg H(s, o)$ for all $s \in S$ and $o \in O$ is an initially secure state
  - Description of the initial state, assumed secure
- Axiom 7-4. If for some $s \in S$ and 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

77

Which Objects Can Be Read?

- Suppose $s \in S$ has read $o \in O$. If s can read $o' \in O$, $o' \neq o$, then $I_1(o') \neq I_1(o)$ or $I_2(o') = I_2(o)$.
  - Says s can read only the objects in a single CD within any COI

78

Proof

Assume false. Then

$H(s, o) \wedge H(s, o') \wedge I_1(o') = I_1(o) \wedge I_2(o') \neq I_2(o)$

Assume s read o first. Then $H(s, o)$ when s read o, so by Axiom 7-2, either $I_1(o') \neq I_1(o)$ or $I_2(o') = I_2(o)$, so

$(I_1(o') \neq I_1(o) \vee I_2(o') = I_2(o)) \wedge (I_1(o') = I_1(o) \wedge I_2(o') \neq I_2(o))$

Rearranging terms,

$(I_1(o') \neq I_1(o) \wedge I_2(o') \neq I_2(o) \wedge I_1(o') = I_1(o)) \vee$

$(I_2(o') = I_2(o) \wedge I_2(o') \neq I_2(o) \wedge I_1(o') = I_1(o))$

which is obviously false, contradiction.

79

Lemma

- 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 7-3, 7-4. If $o' \neq o$, theorem immediately gives lemma.

80

COIs and Subjects

- Theorem: Let $c \in C$ and $d \in D$. Suppose there are n objects $o_i \in O$, $1 \leq i \leq n$, such that $l_1(o_i) = d$ 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 7-2). As there are n such CDs, there must be at least n subjects to meet the conditions of the theorem.

81

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 7-5. $I_1(o) = I_1(v(o))$ iff $I_2(o) = I_2(v(o))$

82

Which Objects Can Be Written?

- Axiom 7-6. $s \in S$ can write to $o \in O$ iff the following hold simultaneously
  1. $H(s, o)$
  2. There is no $o' \in O$ with $H(s, o')$, $I_2(o) \neq I_2(o')$, $I_2(o) \neq I_2(v(o))$, $I_2(o') = I_2(v(o))$.
  - Allow writing iff information cannot leak from one subject to another through a mailbox
  - Note handling for sanitized objects

83

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 above situation as (*o*, *o′*)

84

Key Result

- Set of all information flows is

  $\{ (o, o') \mid o \in O \wedge o' \in O \wedge I_2(o) = I_2(o') \vee I_2(o) = I_2(v(o)) \}$
- Sketch of proof: Definition gives set of flows:

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

  Axiom 7-6 excludes the following flows:

  $X = \{ (o, o') \mid o \in O \wedge o' \in O \wedge I_2(o) \neq I_2(o') \wedge I_2(o) \neq I_2(v(o)) \}$

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

  $F^* - X = \{(o, o') \mid o \in O \wedge o' \in O \wedge \neg(I_2(o) \neq I_2(o') \wedge I_2(o) \neq I_2(v(o)))\}$

  which is equivalent to the claim.

85

Compare to Bell-LaPadula

- Fundamentally different
  - CW has no security labels, B-LP does
  - CW has notion of past accesses, B-LP 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

86

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

87

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

88

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

89

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

90

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

91

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*

92

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

93

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.

94

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.

95

Deletion

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

96

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.

97

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)

98

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)

99

Compare to Bell-LaPadula

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

100

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

101

ORCON

- 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).

102

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.

103

DAC Fails

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

104

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

105

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)

106

RBAC

- 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.

108

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

109

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)

110

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

111

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:

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

112

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:

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

113

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

114

PURDUE UNIVERSITY

CS526: Information Security
Chris Clifton

September 28, 2004

Noninterference

Problem: *Consistent* Policies

- Policies defined by different organizations
  - Different needs
  - But sometimes subjects/objects overlap
- Can all policies be met?
  - Different categories
    - Build lattice combining them
  - Different security levels
    - Need to be *levels* – thus must be able to order

116

What is *Consistent*?

- Principle of autonomy:
  - Access allowed by security policy of a component must be allowed by composition
- Principle of security:
  - Access denied by security policy of a component must be denied by composition
- Must prove new "composed" policy meets these principles

117

Interference

- Expanded notion of "write"
  - noninterference $\Rightarrow$ "single subject" view of the system
  - Any evidence of another subject acting corresponds to write
- Noninterference Definition – $A,G :| G'$:
  - $G, G' \subseteq$ Subjects, $A \subseteq C$ (commands)
  - $\forall c_S \subseteq C^*, s \in G'$:
    - $proj(s,c_S, \sigma_i) = proj(s,\pi_{G,A}(c_S),\sigma_i)$
- Security Policy: Set of noninterference assertions
  - Example: How do you prevent write-down?

118

Key Theorem: Unwinding

- Induction property for interference: For policy *r*, non-interference secure if
  - output consistent: Given a command, the output seen under *r* for subjects not granted rights under *r* for any command *c* is the same for any initial states that appear the same to that subject
  - transition consistent: If the view of subjects under *r* of two states is the same, then the view of the system states after the same command applied to both is the same
  - Locally respects *r*: View of state after a command is the same as the view before the command

119