CS52600: Information Security

Midterm Review

18 October 2010

Prof. Chris Clifton

Course Outline

1. Introduction: Role of security, Types of security, Definitions.
2. Access Control Matrix model
3. Protection Models
4. Policy: Risk Analysis, Policy Formation, Role of audit and control.
5. Formal policy models.
6. Information Flow
7. Authentication and Identity
8. Forensics

Midterm. 10/20

10. System Design principles. TCB and security kernel construction, Verification, Certification issues.
11. Network Security. Distributed cooperation and commit. Distributed authentication issues. Routing, flooding, spamming. Firewalls.
12. Audit Mechanisms.
13. Malicious Code: Viruses, Worms, etc.
14. Vulnerability Analysis.
15. Physical threats, operational security, Legal and Societal Issues

Final Exam

2

Basic Components

- Confidentiality
  - Keeping data and resources hidden
- Integrity
  - Data integrity (integrity)
  - Origin integrity (authentication)
- Availability
  - Enabling access to data and resources

3

Policies and Mechanisms

- Policy says what is, and is not, allowed
  - This defines "security" for the site/system/*etc*.
  - Policy definition: Informal? Formal?
- Mechanisms enforce policies
- Composition of policies
  - If policies conflict, discrepancies may create security vulnerabilities

4

Access Control

- State: Status of the system
  - Protection state: subset that deals with protection
- Access Control Matrix
  - Describes protection state
- Formally:
  - Objects O
  - Subjects S
  - Matrix $A \subseteq S \times O$
- Tuple (S, O, A) defines protection states of system

5

Access Control Matrix: Boolean Evaluation Example

	Internal	Local	State University	Long Distance	International
Public	CR T		R		
Student	CR T	CR T	R	R	R
Staff	CR Transfer	CR T	CR T	R	R
Account	CR T	CR T	CR T	CR T	CR T

6

Protection State Transitions

- State $X_i = (S_i, O_i, A_i)$
- Transitions τ_i
  - Single transition $X_i \vdash_{\tau_{i+1}} X_{i+1}$
  - Series of transitions $X \vdash^* Y$
- Access control matrix may change
  - Change command c associated with transition
  - $X_i \vdash_{c_{i+1}(p_{i+1},\ldots,p_{i+1})} X_{i+1}$
- Change command c associated with transition

7

Primitive Commands

- Create Object *o*
  - Adds *o* to objects with no access
  - $S' = S$, $O' = O \cup \{o\}$, $(\forall x \in S')[a'[x,o] = \varnothing]$, $(\forall x \in S')(\forall y \in O)[a'[x,y] = a[x,y]]$
- Create Subject *s*
  - Adds *s* to objects, subjects, sets relevant access control to $\varnothing$
- Enter *r* into *a*[*s*,*o*]
- Delete *r* from *a*[*s*,*o*]
- Destroy subject *s*, destroy object *o*

8

Formally:

- Given
  - initial state $X_0 = (S_0, O_0, A_0)$
  - Set of primitive commands *c*
- Can we reach a state X_n where ∃s,o such that A_n[s,o] includes a right *r* not in A_0[s,o]?
  - If so, the system is not safe

9

Decidability Result *(Harrison, Ruzzo, Ullman)*

- Given a system where each command consists of a single *primitive* command, There exists an algorithm that will determine if a protection system with initial state X_0 is safe with respect to right *r*.
- Proof: determine minimum commands *k* to leak
  - Delete/destroy: Can't leak (or be detected)
  - Create/enter: new subjects/objects "equal", so treat all new subjects as one
  - If *n* rights, leak possible, must be able to leak in $n(|S_0|+1)(|O_0|+1)+1$ commands
- Enumerate all possible to decide

10

Other Results
(most from the same authors)

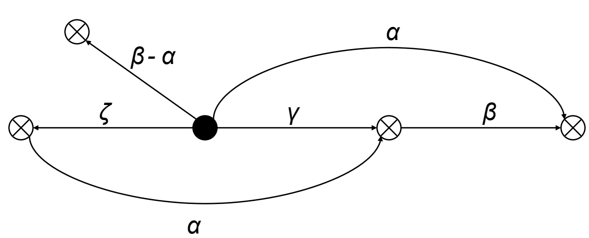
- Set of unsafe systems recursively enumerable
- Without create primitive, safety in P-SPACE
  - Like halting problem reduction, but no unlimited tape
- Without delete/destroy, still undecidable
  - Decidable if at most one condition allowed per command
  - Still holds if delete allowed

11

Take-Grant Protection Model

- System is directed graph
  - Subject: ●
  - Object: ○
  - (labeled) edge: {rights}

  Both: ⊗
- Take rule: if $t \in \gamma$, $\alpha \subseteq \beta$, can add transitive edge
- Grant rule: if $g \in \zeta$, $\alpha \subseteq \gamma$, can add (grant) edge between recipients
- Create, Remove rules



12

Take-Grant Protection Model: Sharing

- Given G_0, can vertex **x** obtain α rights over **y**?
  - Can_share(α,**x**,**y**,G_0) iff $G_0 \vdash^* G_n$ using the above rules and α edge from **x** to **y** in G_n
- *tg-path*: $\mathbf{v}_0,\ldots,\mathbf{v}_n$ where t or g edge between any $\mathbf{v}_i$, $\mathbf{v}_{i+1}$
  - Vertices *tg-connected* if *tg-path* between them
- Theorem: Any two subjects with *tg-path* of length 1 can share rights

13

Theorem: Can_share(α,**x**,**y**,G_0)

- Can_share(α,**x**,**y**,G_0) iff there is an α edge from **x** to **y** in G_0 or if:
  - $\exists$ a vertex **s** $\in G_0$ with an **s** to **y** α edge,
  - $\exists$ a subject **x'** such that **x'**=**x** or **x'** initially spans to **x**,
  - $\exists$ a subject **s'** such that **s'**=**s** or **s'** terminally spans to **s**, and
  - $\exists$ islands $I_1, \ldots, I_n$ such that **x'** $\in I_1$, **s'** $\in I_n$, and there is a bridge from I_j to I_{j+1}
- Proof: If: **x'** grants to **x**, **s'** takes from **s**, otherwise as with subjects
  - Only if: as before, plus object can't give (receive) a right unless someone can take (grant) it
- Corollary: There is an $O(|V|+|E|)$ algorithm to test can_share

15

Theorem: When Theft Possible

- Can_steal(α,**x**,**y**,G_0) iff there is no α edge from **x** to **y** in G_0 and $\exists$ G_1, …, G_n s. t.:
  - There is no α edge from **x** to **y** in G_0 ,
  - $\exists$ subject **x’** such that **x’**=**x** or **x’** initially spans to **x**, and
  - $\exists$ **s** with α edge to **y** in G_0 and can_share(t,**x’**,**s**,G_0)
- Proof:
  - $\Rightarrow$: (easy – build path)
  - $\Leftarrow$: Assume can_steal:
    - No α edge from definition.
    - Can_share(α,**x**,**y**,G_0) from definition: α from **x** to **y** in G_n
    - **s** exists from can_share and Monday’s theorem
    - Can_share(t,**x’**,**s**,G_0): **s** can’t grant α (definition), someone else must get α from **s**, show that this can only be accomplished with take rule

16

Schematic Protection Model

- Key idea: Protection Type τ
  - Label that determines how control rights affect an entity
  - Take-Grant: *subject* and *object* are different protection types
  - Unix file system: File, Directory, ???
- Ticket: Describes a set of rights
  - Entity has set *dom*(**X**) of tickets **Y**/*z* describing **X**’s rights *z* over entities **Y**
- Inert right vs. Control right
  - Inert right doesn’t affect protection state

17

Transferring Rights

- Link predicate: $link_i(\mathbf{X},\mathbf{Y})$
  - conjunction or disjunction of
    - $\mathbf{X}/z \in dom(\mathbf{X})$, $\mathbf{X}/z \in dom(\mathbf{Y})$
    - $\mathbf{Y}/z \in dom(\mathbf{X})$, $\mathbf{Y}/z \in dom(\mathbf{Y})$
    - **true**
  - Determines if **X** and **Y** "connected" to transfer right
  - Example: $link(\mathbf{X},\mathbf{Y}) = \mathbf{Y}/g \in dom(\mathbf{X}) \vee \mathbf{X}/t \in dom(\mathbf{Y})$
- Filter function: conditions on transfer
- Copy $\mathbf{X}/r{:}c$ from **Y** to **Z** allowed iff $\exists i$ such that:
  - $\mathbf{X}/rc \in dom(\mathbf{Y})$
  - $link_i(\mathbf{Y},\mathbf{Z})$
  - $\tau(\mathbf{X})/r{:}c \in filter_i(\tau(\mathbf{Y}), \tau(\mathbf{Z}))$

18

Safety Analysis in SPM

- Idea: derive *maximal state* where changes don't affect analysis
  - Similar to determining max flow
- Theorems:
  - A maximal state exists for every system
  - If parent gives child only rights parent has (conditions somewhat more complex), can easily derive maximal state

19

Typed Access Matrix Model

- Finite set T of types ($TS \subseteq T$ for subjects)
- Protection State: (S, O, τ, A)
  - $\tau: O \rightarrow T$ is a type function
  - Operations same as Harrison-Ruzzo-Ullman except create adds type
- τ is child type iff command creates create subject/object of type τ (otherwise parent)
- If parent/child graph from all commands acyclic, then:
  - Safety is decidable
  - Safety is NP-Hard
  - Safety is polynomial if all commands limited to three parameters

20

Comparing Models

- Expressive Power
  - HRU/Access Control Matrix subsumes Take-Grant
  - HRU subsumes Typed Access Control Matrix
  - SPM subsumes Take-Grant
    - Subject/Object protection types
    - ticket is label on an edge
    - take/grant are control rights
- What about SPM and HRU?
  - SPM has no revocation (delete/destroy)
- HRU without delete/destroy (monotonic HRU)?
  - MTAM subsumes monotonic mono-operational HRU
  - HRU can have create requiring multiple “parents”

21

Extended Schematic Protection Model

- Adds “joint create”: new node has multiple parents
  - Allows more natural representation of sharing between mutually suspicious parties
    - Create joint node for sharing
  - In Take-Grant, SPM, must create two nodes, they interact to share (equivalent power)
- Monotonic ESPM and Monotonic HRU equivalent

22

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?

24

Modeling Secure/Precise: Confidentiality *(Jones and Lipton)*

- 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*

25

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

26

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

27

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 $(r,d,v,v') = (x_t,y_t,z_t,z_{t-1})$

28

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

29

Modeling with Bell-LaPadula: *get-read*

- $r = (\mathit{get}, s, o, \mathbf{r}) \in R^{(1)}$ *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 $\{ (s, o, \mathbf{r}) \}$ added to accesses
    - $(s, o, \mathbf{r})$ must satisfy security properties to reach where it is added
- Similar rules for *get-append*, *execute*, *write*

30

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?*

31

Policies

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

32

Domain-specific Policy Models

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

33

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

34

Information Flow

- Information Flow: Where information can move in the system
- How does this relate to confidentiality policy?
  - Confidentiality: What subjects can see what objects
  - Flow: Controls what subjects actually see
- Variable *x* holds information classified *S*
  - $\underline{x}$, information flow class of *x*, is *S*
- Confidentiality specifies what is allowed
- Information flow describes how this is enforced

35

Formal Definition

- Problem: capturing *all* information flow
  - Files
  - Memory
  - Page faults
  - CPU use
  - ?
- Definition: Based on *entropy*
  - Flow from *x* to *y* (times *s* to *t*) if $H(x_s \mid y_t) < H(x_s \mid y_s)$

36

How do we Manage Information Flow?

- Information flow policy
  - Captures security levels
  - Often based on *confinement*
  - Principles: Reflexivity, transitivity
- Compiler-based mechanisms
  - Track potential flow
  - Enforce legality of flows
- Execution-based mechanisms
  - Track flow at runtime
  - Validate correct

37

Confinement

- Confinement Problem
  - Prevent a server from leaking confidential information
- Covert Channel
  - Path of communication not designed as communication path
- Transitive Confinement
  - If a confined process invokes a second process, invokee must be as confined as invoker

38

Isolation

- Virtual machine
  - Simulates hardware of an (abstract?) machine
  - Process confined to virtual machine
    - Simulator ensures confinement to VM
  - Real example: IBM VM/SP
    - Each user gets “their own” IBM 370
- Sandbox
  - Environment where actions restricted to those allowed by policy

39

Covert Channels

- Storage channel
  - Uses attribute of shared resource
- Timing channel
  - Uses temporal/ordering relationship of access to shared resource
- Noise in covert channel
  - Noiseless: Resource only available to sender/receiver
  - Noisy: Other subjects can affect resource

40

Modeling Covert Channels

- Noninterference
  - Bell-LaPadula approach
  - All shared resources modeled as subjects/objects
  - Let $\sigma \in \Sigma$ be states. Noninterference secure if $\forall s$ at level $l(s)$ $\exists \equiv: \Sigma \times \Sigma$ such that
    - $\sigma_1 \equiv \sigma_2 \Rightarrow \text{view}(\sigma_1) = \text{view}(\sigma_2)$
    - $\sigma_1 \equiv \sigma_2 \Rightarrow \text{execution}(i,\sigma_1) \equiv \text{execution}(i,\sigma_2)$
    - if i only contains instructions from subjects dominating s, $\text{view}(\text{execution}(i, \sigma)) = \text{view}(\sigma)$
- Information Flow analysis
  - Again model all shared resources

41

Test Taking Hints

- Open book/notes
  - Pretty much any non-electronic aid allowed
- See old copies of my exams (and solutions) at my web site
  - CS 526
  - CS 541
  - CS 603
- Time will be tight
  - Suggested “time on question” provided

42

