January 28, 2014

- Bell-LaPadula
  - Informally
  - Formally
  - Example Instantiation

Confidentiality Policy

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

Bell-LaPadula Model, Step 1

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

Example

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

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

Reading Information

- Information flows *up*, not *down*
  - "Reads up" disallowed, "reads down" allowed
- Simple Security Condition (Step 1)
  - Subject s can read object o iff, $L(o) \leq L(s)$ and s has permission to read o
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no reads up" rule

Writing Information

- Information flows up, not down
  - "Writes up" allowed, "writes down" disallowed
- *-Property (Step 1)
  - Subject s can write object o iff $L(s) \le L(o)$ and s has permission to write o
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no writes down" rule

Basic Security Theorem, Step 1

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

Bell-LaPadula Model, Step 2

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

Levels and Lattices

- (A, C) *dom* (A', C') iff $A' \le A$ and $C' \subseteq C$
- Examples
  - (Top Secret, {NUC, ASI}) *dom* (Secret, {NUC})
  - (Secret, {NUC, EUR}) *dom* (Confidential,{NUC, EUR})
  - (Top Secret, {NUC}) $\neg$*dom* (Confidential, {EUR})
- Let C be set of classifications, K set of categories. Set of security levels $L = C \times K$, *dom* form lattice
  - $lub(L) = (max(A), C)$
  - $glb(L) = (min(A), \varnothing)$

Levels and Ordering

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

Reading Information

- Information flows *up*, not *down*
  - "Reads up" disallowed, "reads down" allowed
- Simple Security Condition (Step 2)
  - Subject s can read object o iff $L(s)$ *dom* $L(o)$ and s has permission to read o
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no reads up" rule

Writing Information

- Information flows up, not down
  - "Writes up" allowed, "writes down" disallowed
- *-Property (Step 2)
  - Subject s can write object o iff $L(o)$ *dom* $L(s)$ and s has permission to write o
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no writes down" rule

Basic Security Theorem, Step 2

- If a system is initially in a secure state, and every transition of the system satisfies the simple security condition, step 2, and the *-property, step 2, then every state of the system is secure
  - Proof: induct on the number of transitions
  - In actual Basic Security Theorem, discretionary access control treated as third property, and simple security property and *-property phrased to eliminate discretionary part of the definitions — but simpler to express the way done here.

Problem

- Colonel has (Secret, {NUC, EUR}) clearance
- Major has (Secret, {EUR}) clearance
  - Major can talk to colonel ("write up" or "read down")
  - Colonel cannot talk to major ("read up" or "write down")
- Clearly absurd!

Solution

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

Formal Model

- Allows us to reason precisely about the model
- Provides a formalism to validate systems against

Formal Model Definitions

- S subjects, O objects, P rights
  - Defined rights: r read, a write, w read/write, e empty
- M set of possible access control matrices
- C set of clearances/classifications, K set of categories, $L = C \times K$ set of security levels
- $F = \{ (f_s, f_o, f_c) \}$
  - $f_s(s)$ maximum security level of subject s
  - $f_c(s)$ current security level of subject s
  - $f_o(o)$ security level of object o

More Definitions

- Hierarchy functions $H: O \rightarrow P(O)$
- Requirements
  1. $o_i \neq o_j \Rightarrow h(o_i) \cap h(o_j) = \varnothing$
  2. There is no set $\{ o_1, \ldots, o_k \} \subseteq O$ such that, for $i = 1, \ldots, k$, $o_{i+1} \in h(o_i)$ and $o_{k+1} = o_1$.
- Example
  - Tree hierarchy; take $h(o)$ to be the set of children of o
  - No two objects have any common children (#1)
  - There are no loops in the tree (#2)

States and Requests

- V set of states
  - Each state is (b, m, f, h)
    - b is like m, but excludes rights not allowed by f
- R set of requests for access
- D set of outcomes
  - $\underline{\mathrm{y}}$ allowed, $\underline{\mathrm{n}}$ not allowed, $\underline{\mathrm{i}}$ illegal, $\underline{\mathrm{o}}$ error
- W set of actions of the system
  - $W \subseteq R \times D \times V \times V$

History

- $X = R^N$ set of sequences of requests
- $Y = D^N$ set of sequences of decisions
- $Z = V^N$ set of sequences of states
- Interpretation
  - At time $t \in N$, system is in state $z_{t-1} \in V$; request $x_t \in R$ causes system to make decision $y_t \in D$, transitioning the system into a (possibly new) state $z_t \in V$
- System representation: $\Sigma(R, D, W, z_0) \in X \times Y \times Z$
  - $(x, y, z) \in \Sigma(R, D, W, z_0)$ iff $(x_t, y_t, z_{t-1}, z_t) \in W$ for all t
  - (x, y, z) called an *appearance* of $\Sigma(R, D, W, z_0)$

Example

- $S = \{ s \}$, $O = \{ o \}$, $P = \{ \underline{\text{r}}, \underline{\text{w}} \}$
- $C = \{ \text{High}, \text{Low} \}$, $K = \{ \text{All} \}$
- For every $f \in F$, either $f_c(s) = (\text{ High}, \{ \text{All} \})$ or $f_c(s) = (\text{ Low}, \{ \text{All} \})$
- Initial State:
  - $b_1 = \{ (s, o, \underline{\text{r}}) \}$, $m_1 \in M$ gives s read access over o, and for $f_1 \in F, f_{c,1}(s) = (\text{High}, \{\text{All}\}), f_{o,1}(o) = (\text{Low}, \{\text{All}\})$
  - Call this state $v_0 = (b_1, m_1, f_1, h_1) \in V$.

First Transition

- Now suppose in state v_0: $S = \{ s, s' \}$
- Suppose $f_{c,1}(s') = (\text{Low}, \{\text{All}\})$
- $m_1 \in M$ gives s and s' read access over o
- As s' not written to o, $b_1 = \{ (s, o, \underline{\text{r}}) \}$
- $z_0 = v_0$; if s' requests r_1 to write to o:
  - System decides $d_1 = \underline{\text{y}}$
  - New state $v_1 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, \underline{\text{r}}), (s', o, \underline{\text{w}}) \}$
  - Here, $x = (r_1)$, $y = (\underline{\text{y}})$, $z = (v_0, v_1)$

Second Transition

- Current state $v_1 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, \underline{\mathrm{r}}), (s', o, \underline{\mathrm{w}}) \}$
  - $f_{c,1}(s) = (\text{High}, \{ \text{All} \}), f_{o,1}(o) = (\text{Low}, \{ \text{All} \})$
- s requests r_2 to write to o:
  - System decides $d_2 = \underline{\mathrm{n}}$ (as $f_{c,1}(s)$ *dom* $f_{o,1}(o)$)
  - New state $v_2 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, \underline{\mathrm{r}}), (s', o, \underline{\mathrm{w}}) \}$
  - So, $x = (r_1, r_2)$, $y = (\underline{\mathrm{y}}, \underline{\mathrm{n}})$, $z = (v_0, v_1, v_2)$, where $v_2 = v_1$

Basic Security Theorem

- Define action, secure formally
  - Using a bit of foreshadowing for “secure”
- Restate properties formally
  - Simple security condition
  - *-property
  - Discretionary security property
- State conditions for properties to hold
- State Basic Security Theorem

Action

- A request and decision that causes the system to move from one state to another
  - Final state may be the same as initial state
- $(r, d, v, v') \in R \times D \times V \times V$ is an *action* of $\Sigma(R, D, W, z_0)$ iff there is an $(x, y, z) \in \Sigma(R, D, W, z_0)$ and a $t \in N$ such that $(r, d, v, v') = (x_t, y_t, z_{t-1}, z_t)$
  - Request r made when system in state v; decision d moves system into (possibly the same) state v'
  - Correspondence with (x_t, y_t, z_{t-1}, z_t) makes states, requests, part of a sequence

Simple Security Condition

- $(s, o, p) \in S \times O \times P$ satisfies the simple security condition relative to f (written *ssc rel f*) iff one of the following holds:
  1. $p = \underline{\mathrm{e}}$ or $p = \underline{\mathrm{a}}$
  2. $p = \underline{\mathrm{r}}$ or $p = \underline{\mathrm{w}}$ and $f_s(s)$ *dom* $f_o(o)$
- Holds vacuously if rights do not involve reading
- If all elements of b satisfy *ssc rel f*, then state satisfies simple security condition
- If all states satisfy simple security condition, system satisfies simple security condition

Necessary and Sufficient

- $\Sigma(R, D, W, z_0)$ satisfies the simple security condition for any secure state z_0 iff for every action $(r, d, (b, m, f, h), (b', m', f', h'))$, W satisfies
  - Every $(s, o, p) \in b' - b$ satisfies *ssc rel f*
  - Every $(s, o, p) \in b$ that does not satisfy *ssc rel f* is not in b'
- Note: "secure" means z_0 satisfies *ssc rel f*
- First says every (s, o, p) added satisfies *ssc rel f*; second says any (s, o, p) in b that does not satisfy *ssc rel f* is deleted

*-Property

- $b(s: p_1, \ldots, p_n)$ set of all objects that s has $p_1, \ldots, p_n$ access to
- State (b, m, f, h) satisfies the *-property iff for each $s \in S$ the following hold:
  1. $b(s: \underline{\mathrm{a}}) \neq \varnothing \Rightarrow [\forall o \in b(s: \underline{\mathrm{a}})\ [\ f_o(o)\ dom\ f_c(s)\ ]\ ]$
  2. $b(s: \underline{\mathrm{w}}) \neq \varnothing \Rightarrow [\forall o \in b(s: \underline{\mathrm{w}})\ [\ f_o(o) = f_c(s)\ ]\ ]$
  3. $b(s: \underline{\mathrm{r}}) \neq \varnothing \Rightarrow [\forall o \in b(s: \underline{\mathrm{r}})\ [\ f_c(s)\ dom\ f_o(o)\ ]\ ]$
- Idea: for writing, object dominates subject; for reading, subject dominates object

*-Property

- If all states satisfy simple security condition, system satisfies simple security condition
- If a subset S' of subjects satisfy *-property, then *-property satisfied relative to $S' \subseteq S$
- Note: tempting to conclude that *-property includes simple security condition, but this is false
  - See condition placed on w right for each

Necessary and Sufficient

- $\Sigma(R, D, W, z_0)$ satisfies the *-property relative to $S' \subseteq S$ for any secure state z_0 iff for every action $(r, d, (b, m, f, h), (b', m', f', h'))$, W satisfies the following for every $s \in S'$
  - Every $(s, o, p) \in b' - b$ satisfies the *-property relative to S'
  - Every $(s, o, p) \in b$ that does not satisfy the *-property relative to S' is not in b'
- Note: "secure" means z_0 satisfies *-property relative to S'
- First says every (s, o, p) added satisfies the *-property relative to S'; second says any (s, o, p) in b that does not satisfy the *-property relative to S' is deleted

Discretionary Security Property

- State (b, m, f, h) satisfies the discretionary security property iff, for each $(s, o, p) \in b$, then $p \in m[s, o]$
- Idea: if s can read o, then it must have rights to do so in the access control matrix m
- This is the discretionary access control part of the model
  - The other two properties are the mandatory access control parts of the model

Necessary and Sufficient

- $\Sigma(R, D, W, z_0)$ satisfies the ds-property for any secure state z_0 iff, for every action $(r, d, (b, m, f, h), (b', m', f', h'))$, W satisfies:
  - Every $(s, o, p) \in b' - b$ satisfies the ds-property
  - Every $(s, o, p) \in b$ that does not satisfy the ds-property is not in b
- Note: "secure" means z_0 satisfies ds-property
- First says every (s, o, p) added satisfies the ds-property; second says any (s, o, p) in b that does not satisfy the *-property is deleted

Secure

- A system is secure iff it satisfies:
  - Simple security condition
  - *-property
  - Discretionary security property
- A state meeting these three properties is also said to be secure

Basic Security Theorem

- $\Sigma(R, D, W, z_0)$ is a secure system if z_0 is a secure state and W satisfies the conditions for the preceding three theorems
  – The theorems are on the slides titled "Necessary and Sufficient"

Rule

- $\rho: R \times V \rightarrow D \times V$
- Takes a state and a request, returns a decision and a (possibly new) state
- Rule ρ *ssc-preserving* if for all $(r, v) \in R \times V$ and v satisfying *ssc rel f*, $\rho(r, v) = (d, v')$ means that v' satisfies *ssc rel f′*.
  - Similar definitions for *-property, ds-property
  - If rule meets all 3 conditions, it is *security-preserving*

Unambiguous Rule Selection

- Problem: multiple rules may apply to a request in a state
  - if two rules act on a read request in state v …
- Solution: define relation $W(\omega)$ for a set of rules $\omega = \{ \rho_1, \ldots, \rho_m \}$ such that a state $(r, d, v, v') \in W(\omega)$ iff either
  - $d = \underline{\text{i}}$; or
  - for exactly one integer j, $\rho_j(r, v) = (d, v')$
- Either request is illegal, or only one rule applies

Rules Preserving *SSC*

- Let ω be set of *ssc*-preserving rules. Let state z_0 satisfy simple security condition. Then $\Sigma(R, D, W(\omega), z_0)$ satisfies simple security condition
  - Proof: by contradiction.
    - Choose $(x, y, z) \in \Sigma(R, D, W(\omega), z_0)$ as state not satisfying simple security condition; then choose $t \in N$ such that (x_t, y_t, z_t) is first appearance not meeting simple security condition
    - As $(x_t, y_t, z_t, z_{t-1}) \in W(\omega)$, there is unique rule $\rho \in \omega$ such that $\rho(x_t, z_{t-1}) = (y_t, z_t)$ and $y_t \neq \underline{\mathrm{i}}$.
    - As ρ ssc-preserving, and z_{t-1} satisfies simple security condition, then z_t meets simple security condition, contradiction.

Adding States Preserving *SSC*

- Let $v = (b, m, f, h)$ satisfy simple security condition. Let $(s, o, p) \notin b$, $b' = b \cup \{ (s, o, p) \}$, and $v' = (b', m, f, h)$. Then v' satisfies simple security condition iff:
  1. Either $p = \underline{\text{e}}$ or $p = \underline{\text{a}}$; or
  2. Either $p = \underline{\text{r}}$ or $p = \underline{\text{w}}$, and $f_c(s)$ *dom* $f_o(o)$
  - Proof
    1. Immediate from definition of simple security condition and v' satisfying *ssc rel f*
    2. v' satisfies simple security condition means $f_s(s)$ *dom* $f_o(o)$, and for converse, $(s, o, p) \in b'$ satisfies *ssc rel f*, so v' satisfies simple security condition

Rules, States Preserving *-Property

- Let ω be set of *-property-preserving rules, state z_0 satisfies *-property. Then $\Sigma(R, D, W(\omega), z_0)$ satisfies *-property

Rules, States Preserving ds-Property

- Let ω be set of ds-property-preserving rules, state z_0 satisfies ds-property. Then $\Sigma(R, D, W(\omega), z_0)$ satisfies ds-property

Combining

- Let ρ be a rule and $\rho(r, v) = (d, v')$, where $v = (b, m, f, h)$ and $v' = (b', m', f', h')$. Then:
  1. If $b' \subseteq b, f' = f$, and v satisfies the simple security condition, then v' satisfies the simple security condition
  2. If $b' \subseteq b, f' = f$, and v satisfies the *-property, then v' satisfies the *-property
  3. If $b' \subseteq b$, $m[s, o] \subseteq m'[s, o]$ for all $s \in S$ and $o \in O$, and v satisfies the ds-property, then v' satisfies the ds-property

Proof

1. Suppose v satisfies simple security property.
   a) $b' \subseteq b$ and $(s, o, \underline{r}) \in b'$ implies $(s, o, \underline{r}) \in b$
   b) $b' \subseteq b$ and $(s, o, \underline{w}) \in b'$ implies $(s, o, \underline{w}) \in b$
   c) So $f_c(s)$ *dom* $f_o(o)$
   d) But $f' = f$
   e) Hence $f'_c(s)$ *dom* $f'_o(o)$
   f) So v' satisfies simple security condition

2, 3 proved similarly

Example Instantiation: Multics

- 11 rules affect rights:
  - set to request, release access
  - set to give, remove access to different subject
  - set to create, reclassify objects
  - set to remove objects
  - set to change subject security level
- Set of "trusted" subjects $S_T \subseteq S$
  - *-property not enforced; subjects trusted not to violate
- $\Delta(\rho)$ domain
  - determines if components of request are valid

get-read Rule

- Request $r = (get, s, o, \underline{\mathrm{r}})$
  - s gets (requests) the right to read o
- Rule is $\rho_1(r, v)$:

if $(r \neq \Delta(\rho_1))$ **then** $\rho_1(r, v) = (\underline{\mathrm{i}}, v)$;

else if $(f_s(s)\ dom\ f_o(o)$ **and** $[s \in S_T$ **or** $f_c(s)\ dom\ f_o(o)]$

and $r \in m[s, o])$

then $\rho_1(r, v) = (y, (b \cup \{\ (s, o, \underline{\mathrm{r}})\ \}, m, f, h))$;

else $\rho_1(r, v) = (\underline{\mathrm{n}}, v)$;

Security of Rule

- The get-read rule preserves the simple security condition, the *-property, and the ds-property
  - Proof
    - Let v satisfy all conditions. Let $\rho_1(r, v) = (d, v')$. If $v' = v$, result is trivial. So let $v' = (b \cup \{ (s_2, o, \underline{\mathrm{r}}) \}, m, f, h)$.

Proof

- Consider the simple security condition.
  - From the choice of v', either $b' - b = \varnothing$ or $\{ (s_2, o, \underline{r}) \}$
  - If $b' - b = \varnothing$, then $\{ (s_2, o, \underline{r}) \} \in b$, so $v = v'$, proving that v' satisfies the simple security condition.
  - If $b' - b = \{ (s_2, o, \underline{r}) \}$, because the *get-read* rule requires that $f_s(s)$ *dom* $f_o(o)$, an earlier result says that v' satisfies the simple security condition.

Proof

- Consider the *-property.
  - Either $s_2 \in S_T$ or $f_c(s)$ *dom* $f_o(o)$ from the definition of *get-read*
  - If $s_2 \in S_T$, then s_2 is trusted, so *-property holds by definition of trusted and S_T.
  - If $f_c(s)$ *dom* $f_o(o)$, an earlier result says that v' satisfies the simple security condition.

Proof

- Consider the discretionary security property.
  - Conditions in the *get-read* rule require $\underline{\mathrm{r}} \in m[s, o]$ and either $b' - b = \varnothing$ or $\{ (s_2, o, \underline{\mathrm{r}}) \}$
  - If $b' - b = \varnothing$, then $\{ (s_2, o, \underline{\mathrm{r}}) \} \in b$, so $v = v'$, proving that v' satisfies the simple security condition.
  - If $b' - b = \{ (s_2, o, \underline{\mathrm{r}}) \}$, then $\{ (s_2, o, \underline{\mathrm{r}}) \} \notin b$, an earlier result says that v' satisfies the ds-property.