Chapter 5: Confidentiality Policies

- Overview
  - What is a confidentiality model
- Bell-LaPadula Model
  - General idea
  - Informal description of rules
  - Formal description of rules
- Tranquility
- Controversy
  - †-property
  - System Z
Overview

- **Bell-LaPadula**
  - Informally
  - Formally
  - Example Instantiation
- **Tranquility**
- **Controversy**
  - System Z
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

<table>
<thead>
<tr>
<th>security level</th>
<th>subject</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Secret</td>
<td>Tamara</td>
<td>Personnel Files</td>
</tr>
<tr>
<td>Secret</td>
<td>Samuel</td>
<td>E-Mail Files</td>
</tr>
<tr>
<td>Confidential</td>
<td>Claire</td>
<td>Activity Logs</td>
</tr>
<tr>
<td>Unclassified</td>
<td>Ulaleyl</td>
<td>Telephone Lists</td>
</tr>
</tbody>
</table>

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

• Information flows \textit{up}, not \textit{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 $L(s) \leq L(o)$ 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

- \((A, C) \text{ dom } (A', C')\) iff \(A' \leq A\) and \(C' \subseteq C\)

Examples
- (Top Secret, \{NUC, ASI\}) dom (Secret, \{NUC\})
- (Secret, \{NUC, EUR\}) dom (Confidential, \{NUC, EUR\})
- (Top Secret, \{NUC\}) \(\neg\)dom (Confidential, \{EUR\})

- Let \(C\) be set of classifications, \(K\) set of categories. Set of security levels \(L = C \times K\), dom form lattice
  - \(\text{lub}(L) = (\text{max}(A), C)\)
  - \(\text{glb}(L) = (\text{min}(A), \emptyset)\)
Levels and Ordering

• Security levels partially ordered
  – Any pair of security levels may (or may not) be related by \textit{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 $L(s) \text{ dom } L(o)$
    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 $\text{dom } L(o) \subseteq L(s)$ 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
  – $\text{maxlevel}(s) \; \text{dom} \; \text{curlevel}(s)$

• Example
  – Treat Major as an object (Colonel is writing to him/her)
  – Colonel has $\text{maxlevel}$ (Secret, \{ NUC, EUR \})
  – Colonel sets $\text{curlevel}$ to (Secret, \{ EUR \})
  – Now $L(\text{Major}) \; \text{dom} \; \text{curlevel}(\text{Colonel})$
    • Colonel can write to Major without violating “no writes down”
  – Does $L(s)$ mean $\text{curlevel}(s)$ or $\text{maxlevel}(s)$?
    • Formally, we need a more precise notation
DG/UX System

- Provides mandatory access controls
  - MAC label identifies security level
  - Default labels, but can define others
- Initially
  - Subjects assigned MAC label of parent
    - Initial label assigned to user, kept in Authorization and Authentication database
  - Object assigned label at creation
    - Explicit labels stored as part of attributes
    - Implicit labels determined from parent directory
## MAC Regions

### Hierarchy levels

| VP–1 | Site executables |
| VP–2 | Trusted data |
| VP–3 | Executables not part of the TCB |
| VP–4 | Executables part of the TCB |
| VP–5 | Reserved for future use |

### Categories

| A&A database, audit | Administrative Region |
| User data and applications | User Region |

**IMPL_HI** is “maximum” (least upper bound) of all levels  
**IMPL_LO** is “minimum” (greatest lower bound) of all levels
Directory Problem

• Process $p$ at MAC_A tries to create file /tmp/x
• /tmp/x exists but has MAC label MAC_B
  – Assume MAC_B dom MAC_A
• Create fails
  – Now $p$ knows a file named $x$ with a higher label exists
• Fix: only programs with same MAC label as directory can create files in the directory
  – Now compilation won’t work, mail can’t be delivered
Multilevel Directory

- Directory with a set of subdirectories, one per label
  - Not normally visible to user
  - p creating /tmp/x actually creates /tmp/d/x where d is directory corresponding to MAC_A
  - All p’s references to /tmp go to /tmp/d
- p cd’s to /tmp/a, then to ..
  - System call stat(“.”, &buf) returns inode number of real directory
  - System call dg_stat(“.”, &buf) returns inode of /tmp
Object Labels

- Requirement: every file system object must have MAC label
  1. Roots of file systems have explicit MAC labels
     - If mounted file system has no label, it gets label of mount point
  2. Object with implicit MAC label inherits label of parent
Object Labels

- Problem: object has two names
  - /x/y/z, /a/b/c refer to same object
  - y has explicit label IMPL_HI
  - b has explicit label IMPL_B

- Case 1: hard link created while file system on DG/UX system, so …

3. Creating hard link requires explicit label
   - If implicit, label made explicit
   - Moving a file makes label explicit
Object Labels

- Case 2: hard link exists when file system mounted
  - No objects on paths have explicit labels: paths have same implicit labels
  - An object on path acquires an explicit label: implicit label of child must be preserved

so …

4. Change to directory label makes child labels explicit before the change
Object Labels

• Symbolic links are files, and treated as such, so …

5. When resolving symbolic link, label of object is label of target of the link

• System needs access to the symbolic link itself
Using MAC Labels

- Simple security condition implemented
- *-property not fully implemented
  - Process MAC must equal object MAC
  - Writing allowed only at same security level
- Overly restrictive in practice
MAC Tuples

- Up to 3 MAC ranges (one per region)
- MAC range is a set of labels with upper, lower bound
  - Upper bound must dominate lower bound of range
- Examples
  1. [(Secret, {NUC}), (Top Secret, {NUC})]
  2. [(Secret, ∅), (Top Secret, {NUC, EUR, ASI})]
  3. [(Confidential, {ASI}), (Secret, {NUC, ASI})]
MAC Ranges

1. \[ (\text{Secret}, \{\text{NUC}\}), (\text{Top Secret}, \{\text{NUC}\})] \]
2. \[ (\text{Secret}, \emptyset), (\text{Top Secret}, \{\text{NUC, EUR, ASI}\})] \]
3. \[ (\text{Confidential}, \{\text{ASI}\}), (\text{Secret}, \{\text{NUC, ASI}\})] \]
   - \( (\text{Top Secret}, \{\text{NUC}\}) \) in ranges 1, 2
   - \( (\text{Secret,} \{\text{NUC, ASI}\}) \) in ranges 2, 3
   - \[ (\text{Secret,} \{\text{ASI}\}), (\text{Top Secret,} \{\text{EUR}\})] \) not valid range
     - as \( (\text{Top Secret,} \{\text{EUR}\}) \not\equiv \text{dom} \ (\text{Secret,} \{\text{ASI}\}) \)
Objects and Tuples

• Objects must have MAC labels
  – May also have MAC label
  – If both, tuple overrides label

• Example
  – Paper has MAC range:
    
    $$[(\text{Secret}, \{\text{EUR}\}), (\text{Top Secret}, \{\text{NUC, EUR}\})]$$
MAC Tuples

• Process can read object when:
  – Object MAC range \((lr, hr)\); process MAC label \(pl\)
  – \(pl \ dom \ hr\)
    • Process MAC label grants read access to upper bound of range

• Example
  – Peter, with label \((\text{Secret, \{EUR\}})\), cannot read paper
    • \((\text{Top Secret, \{NUC, EUR\}}) \ dom \ (\text{Secret, \{EUR\}})\)
  – Paul, with label \((\text{Top Secret, \{NUC, EUR, ASI\}})\) can read paper
    • \((\text{Top Secret, \{NUC, EUR, ASI\}}) \ dom \ (\text{Top Secret, \{NUC, EUR\}})\)
MAC Tuples

- Process can write object when:
  - Object MAC range \((lr, hr)\); process MAC label \(pl\)
  - \(pl \in (lr, hr)\)
    - Process MAC label grants write access to any label in range

- Example
  - Peter, with label \((\text{Secret}, \{\text{EUR}\})\), can write paper
    - \((\text{Top Secret}, \{\text{NUC, EUR}\}) \text{ dom } (\text{Secret}, \{\text{EUR}\})\) and \((\text{Secret}, \{\text{EUR}\}) \text{ dom } (\text{Secret}, \{\text{EUR}\})\)
  - Paul, with label \((\text{Top Secret}, \{\text{NUC, EUR, ASI}\})\), cannot read paper
    - \((\text{Top Secret}, \{\text{NUC, EUR, ASI}\}) \text{ dom } (\text{Top Secret}, \{\text{NUC, EUR}\})\)
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. $o_i \neq o_j \Rightarrow h(o_i) \cap h(o_j) = \emptyset$
  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
  - \( y \) allowed, \( n \) not allowed, \( i \) illegal, \( o \) 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

- $S = \{ s \}$, $O = \{ o \}$, $P = \{ r, w \}$
- $C = \{ \text{High, Low} \}$, $K = \{ \text{All} \}$
- For every $f \in F$, either $f_c(s) = (\text{High}, \{ \text{All} \})$ or $f_c(s) = (\text{Low}, \{ \text{All} \})$
- Initial State:
  - $b_1 = \{ (s, o, r) \}$, $m_1 \in M$ 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 $o$, $b_1 = \{ (s, o, r) \}$
- $z_0 = v_0$; if $s'$ requests $r_1$ to write to $o$:
  - System decides $d_1 = y$
  - New state $v_1 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, r), (s', o, w) \}$
  - Here, $x = (r_1), y = (y), z = (v_0, v_1)$
Second Transition

- Current state $v_1 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, r), (s', o, w) \}$
  - $f_{c,1}(s) = (\text{High}, \{ \text{All} \}), f_{o,1}(o) = (\text{Low}, \{ \text{All} \})$
- $s'$ requests $r_2$ to write to $o$:
  - System decides $d_2 = n$ (as $f_{c,1}(s) \ dom f_{o,1}(o)$)
  - New state $v_2 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, r), (s', o, w) \}$
  - So, $x = (r_1, r_2)$, $y = (y, n)$, $z = (v_0, v_1, v_2)$, 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 \((r, d, v, v') = (x_t, y_t, z_t, z_{t-1})\)
  - Request \(r\) made when system in state \(v\); decision \(d\) moves system into (possibly the same) state \(v'\)
  - Correspondence with \((x_t, y_t, z_t, z_{t-1})\) 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. \(p = e\) or \(p = a\)
  2. \(p = r\) or \(p = w\) and \(f(s) \ \text{dom} \ f(o)\)

- 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. $b(s: a) \neq \emptyset \Rightarrow [\forall o \in b(s: a) [ f_o(o) \in dom f_c(s) ] ]$
  2. $b(s: w) \neq \emptyset \Rightarrow [\forall o \in b(s: w) [ f_o(o) = f_c(s) ] ]$
  3. $b(s: r) \neq \emptyset \Rightarrow [\forall o \in b(s: r) [ f_c(s) \in dom f_o(o) ] ]$
- 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 $\rho$ **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
  – $d = i$; 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 $\omega$ 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 $y_t \neq i$. 
    - As $\rho$ $ssc$-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 \( p = e \) or \( p = a \); or
2. Either \( p = r \) or \( p = w \), and \( f_c(s) \in dom f_o(o) \)

- Proof

  1. Immediate from definition of simple security condition and \(v'\) satisfying \(ssc rel f\)

  2. \(v'\) satisfies simple security condition means \( f_c(s) \in dom f_o(o) \), and for converse, \((s, o, p) \in b'\) satisfies \(ssc rel f\), so \(v'\) satisfies simple security condition
Rules, States Preserving *-Property

- Let $\omega$ be set of *-property-preserving rules, state $z_0$ satisfies *-property. Then $\Sigma(R, D, W(\omega), z_0)$ satisfies *-property
- Let $v = (b, m, f, h)$ satisfy *-property. Let $(s, o, p) \notin b$, $b' = b \cup \{(s, o, p)\}$, and $v' = (b', m, f, h)$. Then $v'$ satisfies *-property iff one of the following holds:
  1. $p = e$ or $p = a$
  2. $p = r$ or $p = w$ and $f_c(s) \text{ dom } f_o(o)$
Rules, States Preserving ds-Property

- Let $\omega$ be set of ds-property-preserving rules, state $z_0$ satisfies ds-property. Then $\Sigma(R, D, W(\omega), z_0)$ satisfies ds-property.
- Let $v = (b, m, f, h)$ satisfy ds-property. Let $(s, o, p) \not\in b$, $b' = b \cup \{(s, o, p)\}$, and $v' = (b', m, f, h)$. Then $v'$ satisfies ds-property iff $p \in m[s, o]$. 
Combining

- Let $\rho$ 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 \( (s, o, r) \in b' \) implies \( (s, o, r) \in b \)
   b) \( b' \subseteq b \) and \( (s, o, w) \in b' \) implies \( (s, o, w) \in b \)
   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 $r = (get, s, o, r)$
  – $s$ gets (requests) the right to read $o$
• Rule is $\rho_1(r, v)$:
  
  \[
  \begin{align*}
  \text{if } (r \neq \Delta(\rho_1)) \text{ then } \\
  \rho_1(r, v) &= (i, v); \\
  \text{else if } (f_s(s) \text{ dom } f_o(o) \text{ and } [s \in S_T \text{ or } f_c(s) \text{ dom } f_o(o)]) \\
  \text{and } r \in m[s, o]) \text{ then } \\
  \rho_1(r, v) &= (y, (b \cup \{ (s, o, r) \}, m, f, h)); \\
  \text{else } \rho_1(r, v) &= (n, v);
  \end{align*}
  \]
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 \( v' = (b \cup \{(s_2, o, r)\}, m, f, h) \).
Proof

- Consider the simple security condition.

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

• Consider the \(*\)-property.
  – Either \(s_2 \in S_T\) or \(f_c(s) \text{ dom } f_o(o)\) 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 \(f_c(s) \text{ dom } f_o(o)\), an earlier result says that \(\nu\) satisfies the simple security condition.
Proof

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

• Request $r = (s_1, \text{give}, s_2, o, r)$
  – $s_1$ gives (request to give) $s_2$ the (discretionary) right to read $o$
  – Rule: can be done if giver can alter parent of object
    • If object or parent is root of hierarchy, special authorization required

• Useful definitions
  – $\text{root}(o)$: root object of hierarchy $h$ containing $o$
  – $\text{parent}(o)$: parent of $o$ in $h$ (so $o \in h(\text{parent}(o)))$
  – $\text{canallow}(s, o, v)$: $s$ specially authorized to grant access when object or parent of object is root of hierarchy
  – $m \land m[s, o] \leftarrow r$: access control matrix $m$ with $r$ added to $m[s, o]$
give-read Rule

- Rule is $\rho_6(r, v)$:

  \[
  \text{if } (r \neq \Delta(\rho_6)) \text{ then } \rho_6(r, v) = (i, v);
  \]

  \[
  \text{else if } ([o \neq \text{root}(o) \text{ and } \text{parent}(o) \neq \text{root}(o) \text{ and } \text{parent}(o) \in b(s_1:w)] \text{ or }
  \]

  \[
  [\text{parent}(o) = \text{root}(o) \text{ and } \text{canallow}(s_1, o, v)] \text{ or }
  \]

  \[
  [o = \text{root}(o) \text{ and } \text{canallow}(s_1, o, v)]
  \]

  \[
  \text{then } \rho_6(r, v) = (y, (b, m \land m[s_2, o] \leftarrow r, f, h));
  \]

  \[
  \text{else } \rho_1(r, v) = (n, v);
  \]
Security of Rule

• The *give-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 \( v' = (b, m[s_2, o] \leftarrow r, f, h) \). So \( b' = b, f' = f, m[x, y] = m'[x, y] \) for all \( x \in S \) and \( y \in O \) such that \( x \neq s \) and \( y \neq o \), and \( m[s, o] \subseteq m'[s, o] \). Then by earlier result, \( v' \) satisfies the simple security condition, the *-*property, and the ds-property.
Principle of Tranquility

- Raising object’s security level
  - Information once available to some subjects is no longer available
  - Usually assume information has already been accessed, so this does nothing

- Lowering object’s security level
  - The *declassification problem*
  - Essentially, a “write down” violating *-property
  - Solution: define set of trusted subjects that sanitize or remove sensitive information before security level lowered
Types of Tranquility

• **Strong Tranquility**
  – The clearances of subjects, and the classifications of objects, do not change during the lifetime of the system

• **Weak Tranquility**
  – The clearances of subjects, and the classifications of objects, do not change in a way that violates the simple security condition or the *-property during the lifetime of the system
Example

- DG/UX System
  - Only a trusted user (security administrator) can lower object’s security level
  - In general, process MAC labels cannot change
    - If a user wants a new MAC label, needs to initiate new process
    - Cumbersome, so user can be designated as able to change process MAC label within a specified range
Controversy

• McLean:
  – “value of the BST is much overrated since there is a great deal more to security than it captures. Further, what is captured by the BST is so trivial that it is hard to imagine a realistic security model for which it does not hold.”
  – Basis: given assumptions known to be non-secure, BST can prove a non-secure system to be secure
†-Property

• State \((b, m, f, h)\) satisfies the †-property iff for each \(s \in S\) the following hold:
  1. \(b(s: a) \neq \emptyset \Rightarrow [\forall o \in b(s: a) [f_c(s) \text{ dom } f_o(o) ] ]\)
  2. \(b(s: w) \neq \emptyset \Rightarrow [\forall o \in b(s: w) [f_o(o) = f_c(s) ] ]\)
  3. \(b(s: r) \neq \emptyset \Rightarrow [\forall o \in b(s: r) [f_c(s) \text{ dom } f_o(o) ] ]\)

• Idea: for writing, subject dominates object; for reading, subject also dominates object

• Differs from *-property in that the mandatory condition for writing is reversed
  – For *-property, it’s object dominates subject
Analogues

The following two theorems can be proved

• $\Sigma(R, D, W, z_0)$ satisfies the $\dagger$-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 $\dagger$-property relative to $S'$
  - Every $(s, o, p) \in b'$ that does not satisfy the $\dagger$-property relative to $S'$ is not in $b$

• $\Sigma(R, D, W, z_0)$ is a secure system if $z_0$ is a secure state and $W$ satisfies the conditions for the simple security condition, the $\dagger$-property, and the ds-property.
Problem

• This system is \textit{clearly} non-secure!
  – Information flows from higher to lower
    because of the \textdagger-property
Discussion

• Role of Basic Security Theorem is to demonstrate that rules preserve security

• Key question: what is security?
  – Bell-LaPadula defines it in terms of 3 properties (simple security condition, *-property, discretionary security property)
  – Theorems are assertions about these properties
  – Rules describe changes to a particular system instantiating the model
  – Showing system is secure requires proving rules preserve these 3 properties
Rules and Model

- Nature of rules is irrelevant to model
- Model treats “security” as axiomatic
- Policy defines “security”
  - This instantiates the model
  - Policy reflects the requirements of the systems
- McLean’s definition differs from Bell-LaPadula
  - … and is not suitable for a confidentiality policy
- Analysts cannot prove “security” definition is appropriate through the model
System Z

- System supporting weak tranquility
- On any request, system downgrades all subjects and objects to lowest level and adds the requested access permission
  - Let initial state satisfy all 3 properties
  - Successive states also satisfy all 3 properties
- Clearly not secure
  - On first request, everyone can read everything
Reformulation of Secure Action

- Given state that satisfies the 3 properties, the action transforms the system into a state that satisfies these properties and eliminates any accesses present in the transformed state that would violate the property in the initial state, then the action is secure.
- BST holds with these modified versions of the 3 properties.
Reconsider System Z

• Initial state:
  – subject $s$, object $o$
  – $C = \{\text{High, Low}\}$, $K = \{\text{All}\}$

• Take:
  – $f_c(s) = (\text{Low, \{All\}}), f_o(o) = (\text{High, \{All\}})$
  – $m[s, o] = \{w\}$, and $b = \{(s, o, w)\}$.

• $s$ requests $r$ access to $o$

• Now:
  – $f'_o(o) = (\text{Low, \{All\}})$
  – $(s, o, r) \in b', m'[s, o] = \{r, w\}$
Non-Secure System Z

• As \((s, o, r) \in b' - b\) and \(f'_o(o) \in \text{dom} \ f'_c(s)\), access added that was illegal in previous state
  – Under the new version of the Basic Security Theorem, System Z is not secure
  – Under the old version of the Basic Security Theorem, as \(f'_c(s) = f'_o(o)\), System Z is secure
Response: What Is Modeling?

- Two types of models
  1. Abstract physical phenomenon to fundamental properties
  2. Begin with axioms and construct a structure to examine the effects of those axioms
- Bell-LaPadula Model developed as a model in the first sense
  - McLean assumes it was developed as a model in the second sense
Reconciling System Z

• Different definitions of security create different results
  – Under one (original definition in Bell-LaPadula Model), System Z is secure
  – Under other (McLean’s definition), System Z is not secure
Key Points

• Confidentiality models restrict flow of information
• Bell-LaPadula models multilevel security
  – Cornerstone of much work in computer security
• Controversy over meaning of security
  – Different definitions produce different results