Confidentiality Policies

ECS 153 Spring Quarter 2021

Module 9
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 are called *security clearance L*(s) for subjects and *security classification L*(o) for objects
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>Ulaley</td>
<td>Telephone Lists</td>
</tr>
</tbody>
</table>

- 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 $L(s) \leq L(o)$ 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\}) \n\text{dom} (Confidential, \{EUR\})

• Let \(C\) be set of classifications, \(K\) set of categories. Set of security levels \(L = C \times K\), dom form lattice
  • \(lub(L) = (\max(A), C)\)
  • \(glb(L) = (\min(A), \emptyset)\)
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 $\text{dom } L(o)$ 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 $L(o) \text{ dom } L(s)$ 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 $L(Major) \ dom \ curlevel(\text{Colonel})$
    • 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
Example: Trusted Solaris

• Provides mandatory access controls
  • Security level represented by sensitivity label
  • Least upper bound of all sensitivity labels of a subject called clearance
  • Default labels ADMIN_HIGH (dominates any other label) and ADMIN_LOW (dominated by any other label)

• $S$ has controlling user $U_S$
  • $S_L$ sensitivity label of subject
  • $\text{privileged}(S, P)$ true if $S$ can override or bypass part of security policy $P$
  • $\text{asserted} (S, P)$ true if $S$ is doing so
Rules

$C_L$ clearance of $S$, $S_L$ sensitivity label of $S$, $U_S$ controlling user of $S$, and $O_L$ sensitivity label of $O$

1. If $\neg privileged(S, \text{"change } S_L\text{"})$, then no sequence of operations can change $S_L$ to a value that it has not previously assumed.

2. If $\neg privileged(S, \text{"change } S_L\text{"})$, then $\neg privileged(S, \text{"change } S_L\text{"})$.

3. If $\neg privileged(S, \text{"change } S_L\text{"})$, then no value of $S_L$ can be outside the clearance of $U_S$.

4. For all subjects $S$, named objects $O$, if $\neg privileged(S, \text{"change } O_L\text{"})$, then no sequence of operations can change $O_L$ to a value that it has not previously assumed.
Rules (con’t)

$C_L$ clearance of $S$, $S_L$ sensitivity label of $S$, $U_S$ controlling user of $S$, and $O_L$ sensitivity label of $O$

5. For all subjects $S$, named objects $O$, if ¬privileged($S$, “override $O$’s mandatory read access control”), then read access to $O$ is granted only if $S_L \text{ dom } O_L$
   • Instantiation of simple security condition

6. For all subjects $S$, named objects $O$, if ¬privileged($S$, “override $O$’s mandatory write access control”), then write access to $O$ is granted only if $O_L \text{ dom } S_L$ and $C_L \text{ dom } O_L$
   • Instantiation of *-property
Initial Assignment of Labels

• Each account is assigned a label range [clearance, minimum]
• On login, Trusted Solaris determines if the session is single-level
  • If clearance = minimum, single level and session gets that label
  • If not, multi-level; user asked to specify clearance for session; must be in the label range
  • In multi-level session, can change to any label in the range of the session clearance to the minimum
Writing

• Allowed when subject, object labels are the same or file is in downgraded directory $D$ with sensitivity label $D_L$ and all the following hold:
  • $S_L \text{ dom } D_L$
  • $S$ has discretionary read, search access to $D$
  • $O_L \text{ dom } S_L$ and $O_L \neq S_L$
  • $S$ has discretionary write access to $O$
  • $C_L \text{ dom } O_L$
• Note: subject cannot read object
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
  - System call stat(".", &buf) returns information about /tmp/d
  - System call mldstat(".", &buf) returns information about /tmp
Labeled Zones

- Used in Trusted Solaris Extensions, various flavors of Linux

- **Zone**: virtual environment tied to a unique label
  - Each process can only access objects in its zone

- **Global zone** encompasses everything on system
  - Its label is ADMIN_HIGH
  - Only system administrators can access this zone

- Each zone has a unique root directory
  - All objects within the zone have that zone’s label
  - Each zone has a unique label
More about Zones

• Can import (mount) file systems from other zones provided:
  • If importing *read-only*, importing zone’s label must dominate imported zone’s label
  • If importing *read-write*, importing zone’s label must equal imported zone’s label
    • So the zones are the same; import unnecessary
  • Labels checked at time of import
• Objects in imported file system retain their labels
Example

• $L_1 \text{ dom } L_2$
• $L_3 \text{ dom } L_2$
• Process in $L_1$ can read any file in the export directory of $L_2$ (assuming discretionary permissions allow it)
• $L_1$, $L_3$ disjoint
  • Do not share any files
• System directories imported from global zone, at ADMIN_LOW
  • So can only be read
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: Trusted Solaris

• Security administrator can provide specific authorization for a user to change the MAC label of a file
  • “downgrade file label” authorization
  • “upgrade file label” authorization

• User requires additional authorization if not the owner of the file
  • “act as file owner” authorization
Principles of Declassification

• Principle of Semantic Consistency
  • As long as semantics of components that do not do declassification do not change, the components can be altered without affecting security

• Principle of Occlusion
  • A declassification operation cannot conceal an *improper* declassification

• Principle of Conservativity
  • Absent any declassification, the system is secure

• Principle of Monotonicity of Release
  • When declassification is performed in an authorized manner by authorized subjects, the system remains secure