Trust in Formal Methods

1. Proof has no errors
   - Bugs in automated theorem provers

2. Preconditions hold in environment in which $S$ is to be used

3. $S$ transformed into executable $S’$ whose actions follow source code
   - Compiler bugs, linker/loader/library problems

4. Hardware executes $S’$ as intended
   - Hardware bugs (Pentium 1001 bug, for example)
Types of Access Control

• Discretionary Access Control (DAC, IBAC)
  • Individual user sets access control mechanism to allow or deny access to an object

• Mandatory Access Control (MAC)
  • System mechanism controls access to object, and individual cannot alter that access

• Originator Controlled Access Control (ORCON, ORGCON)
  • Originator (creator) of information controls who can access information
Types of Mechanisms

- Secure
- Precise
- Broad

Set of reachable states
Set of secure states
Secure, Precise Mechanisms

• Can one devise a procedure for developing a mechanism that is both secure and precise?
  • Consider confidentiality policies only here
  • Integrity policies produce same result

• Program a function with multiple inputs and one output
  • Let $p$ be a function $p: I_1 \times \ldots \times I_n \to R$. Then $p$ is a program with $n$ inputs $i_k \in I_k$, $1 \leq k \leq n$, and one output $r \to R$
Programs and Postulates

• Observability Postulate: the output of a function encodes all available information about its inputs
  • Covert channels considered part of the output

• Example: authentication function
  • Inputs name, password; output Good or Bad
  • If name invalid, immediately print Bad; else access database
  • Problem: time output of Bad, can determine if name valid
  • This means timing is part of output
Protection Mechanism

• Let \( p \) be a function \( p: I_1 \times \ldots \times I_n \rightarrow R \). A protection mechanism \( m \) is a function

\[
m: I_1 \times \ldots \times I_n \rightarrow R \cup E
\]

for which, when \( i_k \in I_k, \ 1 \leq k \leq n \), either

• \( m(i_1, \ldots, i_n) = p(i_1, \ldots, i_n) \) or

• \( m(i_1, \ldots, i_n) \in E \).

• \( E \) is set of error outputs

• In above example, \( E = \{ \text{“Password Database Missing”}, \text{“Password Database Locked”} \} \)
Confidentiality Policy

• Confidentiality policy for program \( p \) says which inputs can be revealed
  • Formally, for \( p: I_1 \times \ldots \times I_n \rightarrow R \), it is a function \( c: I_1 \times \ldots \times I_n \rightarrow A \), where
    \[ A \subseteq I_1 \times \ldots \times I_n \]
    • \( A \) is set of inputs available to observer
  • Security mechanism is function \( m: I_1 \times \ldots \times I_n \rightarrow R \cup E \)
    • \( m \) is secure if and only if \( \exists m': A \rightarrow R \cup E \) such that,
      \[ \forall i_k \in I_k, 1 \leq k \leq n, m(i_1, \ldots, i_n) = m'(c(i_1, \ldots, i_n)) \]
    • \( m \) returns values consistent with \( c \)
Examples

• $c(i_1, \ldots, i_n) = C$, a constant
  • Deny observer any information (output does not vary with inputs)

• $c(i_1, \ldots, i_n) = (i_1, \ldots, i_n)$, and $m' = m$
  • Allow observer full access to information

• $c(i_1, \ldots, i_n) = i_1$
  • Allow observer information about first input but no information about other inputs.
Precision

• Security policy may be over-restrictive
  • Precision measures how over-restrictive

• $m_1, m_2$ distinct protection mechanisms for program $p$ under policy $c$
  • $m_1$ as precise as $m_2$ ($m_1 \approx m_2$) if, for all inputs $i_1, ..., i_n$,
    $$m_2(i_1, ..., i_n) = p(i_1, ..., i_n) \Rightarrow m_1(i_1, ..., i_n) = p(i_1, ..., i_n)$$
  • $m_1$ more precise than $m_2$ ($m_1 \sim m_2$) if there is an input $$(i_1', ..., i_n')$$ such that
    $$m_1(i_1', ..., i_n') = p(i_1', ..., i_n')$$ and
    $$m_2(i_1', ..., i_n') \neq p(i_1', ..., i_n').$$
Combining Mechanisms

- \( m_1, m_2 \) protection mechanisms

- \( m_3 = m_1 \cup m_2 \)
  - For inputs on which \( m_1 \) and \( m_2 \) return same value as \( p \), \( m_3 \) does also; otherwise, \( m_3 \) returns same value as \( m_1 \)

- Theorem: if \( m_1, m_2 \) secure, then \( m_3 \) secure
  - Also, \( m_3 \approx m_1 \) and \( m_3 \approx m_2 \)
  - Follows from definitions of secure, precise, and \( m_3 \)
Existence Theorem

• For any program $p$ and security policy $c$, there exists a precise, secure mechanism $m^*$ such that, for all secure mechanisms $m$ associated with $p$ and $c$, $m^* \approx m$
  • Maximally precise mechanism
  • Ensures security
  • Minimizes number of denials of legitimate actions
Lack of Effective Procedure

• There is no effective procedure that determines a maximally precise, secure mechanism for any policy and program.

  • Sketch of proof: let policy $c$ be constant function, and $p$ compute function $T(x)$. Assume $T(x) = 0$. Consider program $q$, where

    
    $p;$
    
    $\text{if } z = 0 \text{ then } y := 1 \text{ else } y := 2;$
    
    $\text{halt;}$
Rest of Sketch

• $m$ associated with $q$, $y$ value of $m$, $z$ output of $p$ corresponding to $T(x)$
• $\forall x[T(x) = 0] \rightarrow m(x) = 1$
• $\exists x^\prime [T(x^\prime) \neq 0] \rightarrow m(x) = 2$ or $m(x)$ undefined
• If you can determine $m$, you can determine whether $T(x) = 0$ for all $x$
• Determines some information about input (is it 0?)
• Contradicts constancy of $c$.
• Therefore no such procedure exists
Key Points

• Policies describe *what* is allowed
• Mechanisms control *how* policies are enforced
• Trust underlies everything
Outline

• Overview
  • What is a confidentiality model

• Bell-LaPadula Model
  • General idea
  • Informal description of rules
  • Formal description of rules

• Tranquility

• Declassification

• Controversy
  • †-property
  • 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 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 \((\text{clearance}, \text{category set})\)
• Examples
  • \((\text{Top Secret}, \{\text{NUC, EUR, ASI}\})\)
  • \((\text{Confidential}, \{\text{EUR, ASI}\})\)
  • \((\text{Secret}, \{\text{NUC, ASI}\})\)
Levels and Lattices

• \((A, C)\) 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\}) \(\not\) dom (Confidential, \{EUR\})

• Let \(C\) be set of classifications, \(K\) set of categories. Set of security levels \(L = C \times K\), dom form lattice
  • \(lub(L) = (\max(A), C)\)
  • \(glb(L) = (\min(A), \emptyset)\)