Security Policies

Basic Definitions

- **Security policy of a system**: a statement that partitions the states of the system into a set of **secure states** and a set of **nonsecure states**
- **Secure system**: a system that is always in a secure state
- **A breach of security** occurs when a system enters a nonsecure state

Redefine C.I.A.

- Let $X$ be a set of entities and let $I$ be some info
- I has **confidentiality** with respect to $X$ if no member of $X$ can obtain info about $I$
- I has **integrity** with respect to $X$ if all members of $X$ trust $I$
- I has **availability** with respect to $X$ if all members of $X$ can access $I$

Security Policy vs. C.I.A.

- A security policy considers aspects of confidentiality, integrity, and availability
- W.r.t. confidentiality
  - Identify states in which info leakage happens
  - W.r.t. integrity
  - Describes the condition and manner that data can be altered
  - W.r.t. availability
  - Describes what services must be provided

Types of Security Policies

- **Military security policy** (or governmental security policy) primarily concerns confidentiality
- **Commercial security policy** primarily concerns integrity

Security Model

- A security model represents a particular policy, or set of policies
Access Control in Security Policy

- **Discretionary access control (DAC)**
  - An individual user can set an access control mechanism to allow or deny access to an object
  - Also called identity-based access control (IBAC)
- **Mandatory access control (MAC)**
  - A system mechanism controls access to an object
  - Individual users cannot alter that access
  - Also called rule-based access control
- **Originator controlled access control (ORCON)**
  - The creator has the control

Policy Languages

- High-level policy languages
- Low-level policy languages

High-level Policy Language

- Describes constraints placed on entities and actions in a system
- No ambiguity
  - Requires mathematical or programmatic languages
- Ignores implementation issues
  - Just specify

Example 1: Java classes and methods

- Pandey and Hashii defined a security policy for downloaded Java programs
- Format: `deny (s op x) when b`
  - s : x : s creates an instance of class x
  - s op x : s invokes a method x
  - b : a boolean expression
  - Means s cannot "op" on x when b is true
- Policy can be statement like the following:
  - `deny (-| Socket) when (Network.numconns >= 100)`
  - `deny (-Æ file.read) when (file.getfilename() == "/etc/passwd")`

Example 2: DTEL

- **DTEL**: Domain-type enforcement language
- Every subject has a **domain**
- Every object has a **type**
- DTEL constraints the actions that a subject can perform on an object

Case Study using DTEL

- How can we express the mandatory access control of files in a system
  - d_user : domain for ordinary users
  - d_admin : domain for administrative users
  - t_sysbin : type for executable files
  - t_readable: type for readable files
  - t_writable: type for writable files
(cont’d)

Low-level Policy Languages

- Simply commands with a set of inputs or arguments to set or check system constraints
- Example:
  - In X Windows, a user at hood.cs.uoregon.edu can run
    \texttt{xhost ix.cs.uoregon.edu}
    to allow an X client on \texttt{ix} to talk to the X server on \texttt{hood}.

Tripwire

- Question: what does the following mean?
  \texttt{/usr/bin/tripwire +gimpsu012345678-a}
- Hint: log on to \texttt{ix.cs.uoregon.edu} (or other Unix machines) and type:
  \texttt{$ man tripwire}
  to read the manual for \texttt{tripwire}.

Confidentiality Policies

- A \texttt{confidentiality policy} prevents unauthorized information disclosure
  - Also called information flow policy
- Focus on \texttt{Bell-LaPadula Model}
  - Combines MAC and DAC

Goals

- \texttt{Bell-LaPadula Model}
  - Corresponds to military-style security classifications
    - Top secret, secret, confidential, unclassified
  - A subject has a security clearance
    - One of the four above
  - An object has a security classification (or security level)
    - Also one of the four above
  - Goal: prevent read access to objects at a security classification higher than the subject’s clearance
Example

<table>
<thead>
<tr>
<th>Classification</th>
<th>Name(s)</th>
<th>File(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP SECRET (TS)</td>
<td>Ted, Tom</td>
<td>Personal Records</td>
</tr>
<tr>
<td>SECRET (S)</td>
<td>Smith, Sarah</td>
<td>Emails</td>
</tr>
<tr>
<td>CONFIDENTIAL (C)</td>
<td>Cathy, Clarence</td>
<td>Activity Log</td>
</tr>
<tr>
<td>UNCLASSIFIED</td>
<td>Ubi</td>
<td>Phone Directory</td>
</tr>
</tbody>
</table>

Notations (1)

- $L(S) = l_s$: Subject $S$’s security clearance
- $L(O) = l_o$: Object $O$’s security classification
- For all security classifications $l_i$ ($i = 0, \ldots, k-1$), $l_i < l_{i+1}$

Simple Security Condition

preliminary version

- $S$ can read $O$ iff $l_o \leq l_s$ and $S$ has discretionary read access to $O$.

*-Property

preliminary version

- $S$ can write $O$ iff $l_s \leq l_o$ and $S$ has discretionary write access to $O$.

Basic Security Theorem

preliminary version

- $\mathcal{S}$: a system with a secure initial state $\mathcal{S}_0$
- $T$: a set of state transformations
- If every elements of $T$ preserves the simple security condition (preliminary version) and the *-property (preliminary version), then every state $\mathcal{S}(i \geq 0)$ is secure.

Redefined “Security Level”

- By adding the concept of categories
  - Objects are placed in categories
- “Need to know” principle
  - A subject should not access info that is unnecessary for its task
  - So a subject may only access objects in certain categories
- Security level now can also contain category info
  - Expressed as $(L, C)$
  - E.g. William cleared into the level (SECRET, {EUR})
  - E.g. A doc classified as (CONFIDENTIAL, {US})
Notations (2)

• The security level \((L, C)\) dominates the security level \((L', C')\) iff \(L' \leq L\) and \(C' \supseteq C\)
  
  denoted as \((L, C) \triangleright= (L', C')\)
• \((L, C) \not\triangleright (L', C')\) when \((L, C) \triangleright (L', C')\) is false
  
  – Meaning \(L' > L\) OR \(C' \not\supseteq C\)
• \(C(S)\): Subject \(S\)'s category set
• \(C(O)\): Object \(O\)'s category set

Examples

• George : (SECRET, \{ASIA, EUR\})
• Doc A : (CONFIDENTIAL, \{ASIA\})
• Doc B : (SECRET, \{EUR, US\})
• Doc C : (SECRET, \{EUR\})

• George \? Doc A
• George \? Doc B
• George \? Doc C

Simple Security Condition

• \(S\) can read \(O\) iff
  
  \(S\) dom \(O\) and

  \(S\) has discretionary read access to \(O\).

• Can George read Doc A?
  – How about Doc B?

*-Property

• \(S\) can write \(O\) iff
  
  \(O\) dom \(S\) and

  \(S\) has discretionary write access to \(O\).

• Suppose Paul : (SECRET, \{ASIA, EUR, US\})
• Recall Doc B : (SECRET, \{EUR, US\})
• Doc A : (CONFIDENTIAL, \{ASIA\})
• Can Paul write Doc A?
• Can Paul write Doc B?

Basic Security Theorem

• \(\triangleright=\): a system with a secure initial state \(\triangleright= 0\)
• \(T\): a set of state transformations
• If every elements of \(T\) preserves the simple security condition and the \(\triangleright=\)-property, then every state \(\triangleright= i (i \geq 0)\) is secure.

Homework 1

(Due 4/15)

• Chapter 1
  [Page 25] 1.12 : 1, 2, 5
• Chapter 2
  [Page 44] 2.8 : 1, 4
• Chapter 4
  [Page 120] 4.11 : 5, 6, 7

[Also] On a Unix machine, how can Michal use tripwire to monitor whether somebody else has accessed his home directory /home/michal?