Lecture 19: Information Flow
Where we were…

- **Authentication**: mechanisms that bind principals to actions
- **Authorization**: mechanisms that govern whether actions are permitted
- **Audit**: mechanisms that record and review actions
Access Control Policy

- An **access control policy** specifies which of the **operations** associated with any given **object** each **subject** is authorized to perform.

- Expressed as a relation **Auth**:

<table>
<thead>
<tr>
<th>Auth</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dac.tex</td>
</tr>
<tr>
<td>subject</td>
<td>ebirdell</td>
</tr>
<tr>
<td></td>
<td>clarkson</td>
</tr>
<tr>
<td></td>
<td>student</td>
</tr>
</tbody>
</table>

- Illustration of the access control policy with examples of authorized operations for different subjects on specific objects.
Who defines Policies?

• **Discretionary access control (DAC)**
  - **Philosophy:** users have the *discretion* to specify policy themselves
  - Commonly, information belongs to the **owner** of object
  - Access control lists, privilege lists, capabilities

• **Mandatory access control (MAC)**
  - **Philosophy:** central authority *mandates* policy
  - Information belongs to the authority, not to the individual users
  - MLS and BLP, Chinese wall, Clark-Wilson, etc.
Access control for computed data
Scaling to many pieces of data...
Scaling to many users…
Scaling to many interactions…
Information Flows between Principals

- **Channel**: means to communicate information
- **Storage channel**: written by one program and read by another
- **Legitimate channel**: intended for communication between programs
- **Covert channel**: not intended for information transfer yet exploitable for that purpose

Sometimes, we really want to restrict access to information
Information Flow (IF) Policies

• Focus on **information** not objects
• An IF policy specifies **restrictions** on the associated data, and on all its derived data.
• IF policy for confidentiality:
  • Value $\nu$ and all its derived values are allowed to be read only by Alice

Different from the access control policy:
Value $\nu$ is allowed to be read at most by Alice.

• The enforcement mechanism **automatically** deduces the restrictions for derived data.
Information flow policies

Can flow to: Alice

Automatic deduction of policies!
Scaling to many interactions...
Scaling to many interactions...
Labels represent policies

```
Secret, \{nuc, crypto\}

Secret, \{nuc\}
Conf, \{nuc, crypto\}
Secret, \{crypto\}

Conf, \{nuc\}
Secret, \{}\nConf, \{crypto\}

Conf, \{}\n```
Labels represent policies

High

Low
Labels represent policies
Policy Granularity

- Objects can be system principles (files, programs, sockets…)
- Objects can be program variables
Noninterference

[Goguen and Meseguer 1982]

An interpretation of noninterference for a program:

- Changes on H inputs should not cause changes on L outputs.
Noninterference: Example

The program satisfies noninterference!
Noninterference: Example

The program does not satisfy noninterference!
Noninterference: Example

The program does not satisfy noninterference!
Noninterference

- Consider a program $C$.
- Consider two memories $M_1$ and $M_2$, such that
  - they agree on values of variables tagged with $L$:
    $$M_1 =_L M_2.$$ 
  - $M_1$ and $M_2$ may not agree on values of variables tagged with $H$.
- $C(M_i)$ are the observations produced by executing $C$ to termination on initial memory $M_i$:
  - final outputs, or
  - intermediate and final outputs.
- Then, observations tagged with $L$ should be the same:
  $$C(M_1) =_L C(M_2).$$
Noninterference

For a program $C$ and a mapping from variables to labels in \{L, H\}:

$$\forall M_1, M_2: \text{ if } M_1 =_L M_2, \text{ then } C(M_1) =_L C(M_2).$$
Less restrictive than necessary…

\[ h \]

while \( h > 5 \) do

\[ l' := 4 \]

while \( h > 5 \) do

\[ l' := 4 \]
Termination sensitive noninterference

• If
  • $M_1 =_L M_2$,

• then
  • $C$ terminates on $M_1$ iff $C$ terminates on $M_2$, and
  • $C(M_1) =_L C(M_2)$. 
Less restrictive than necessary…

\[ m := \text{Match} \text{(students; grades)} \]

Required to be \( L \).

Wanted to be \( H \)!
More restrictive than necessary...

\[ x := \text{maj}(v_1, v_2, \ldots, v_n) \]

Required to be H.

Wanted to be L!
More restrictive than necessary…

\[ x := \text{Enc}(v; k) \]

Required to be \( H \).

Wanted to be \( L! \)
Declassification

• **What**: specify what information may be declassified
  • e.g., LastFourDigits(credit card number) should be low
  • Partial Equivalence Relation (PER) Model

• **Who**: specify who may declassify information
  • e.g., high object owner can write to low objects
  • Decentralized Label Model

• **Where**: specify which pieces of code may declassify
  • e.g., encryption function can write to low objects
  • Intransitive Noninterference, Reactive Noninterference

• **When**: specify when information may be declassified
  • e.g., software key may be shared after payment has been received
Enforcement Mechanisms

- taint-tracking
- runtime monitoring
- type checking