Lecture 4: Access control (cont.), information flow control

Eran Tromer

Slides credit:
John Mitchell, Stanford
Max Krohn, MIT
Ian Goldberg and Urs Hengartner, University of Waterloo
Multi-Level Security (MLS) Concepts

- **Military security policy**
  - Classification involves sensitivity levels, compartments
  - Do not let classified information leak to unclassified files

- **Group individuals and resources**
  - Use some form of hierarchy to organize policy

- **Other policy concepts**
  - Separation of duty
  - “Chinese Wall” Policy (managing conflicts of interests)
Military security policy

- **Sensitivity levels**
- **Compartments**

Satellite data

- Afghanistan
- Middle East
- Israel

<table>
<thead>
<tr>
<th>Top Secret</th>
<th>Secret</th>
<th>Confidential</th>
<th>Restricted</th>
<th>Unclassified</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Military security policy

Classification of personnel and data
- Class = \langle \text{rank, compartment} \rangle

Dominance relation
- \( D_1 \leq D_2 \) iff \( \text{rank}_1 \leq \text{rank}_2 \)
  and \( \text{compartment}_1 \subseteq \text{compartment}_2 \)

- Example: \( \langle \text{Restricted, Israel} \rangle \leq \langle \text{Secret, Middle East} \rangle \)

Applies to
- Subjects – users or processes
- Objects – documents or resources
Commercial version

Product specifications

Discontinued

In production

OEM

Internal

Proprietary

Public
Generalizing: lattices

- A dominance relationship that is transitive and antisymmetric defines a **lattice**
- For two levels $a$ and $b$, maybe neither $a \geq b$ nor $b \geq a$
- However, for every $a$ and $b$, there is a **lowest upper bound** $u$ for which $u \geq a$ and $u \geq b$, and
  a **greatest lower bound** $l$ for which $a \geq l$ and $b \geq l$
- There are also two elements $U$ and $L$ that dominate/are dominated by all levels
  - In next slide’s example,
    - $U = ("Top Secret", \{"Middle East", "Micronesia"\})$
    - $L = ("Unclassified", \emptyset)$
Example Lattice

(“Top Secret”, {“Middle East”, “Micronesia”}),

(“Top Secret”, {“Middle East”})

(“Secret”, {“Middle East”, “Micronesia”})

(“Secret”, {“Middle East”})

(“Secret”, {“Micronesia”})

(“Unclassified”, ∅)
Bell-LaPadula Confidentiality Model

When is it OK to release information?

Two properties

- No read up
  - A subject S may read object O only if $C(O) \leq C(S)$

- No write down
  - A subject S with read access to O may write object P only if $C(O) \leq C(P)$

You may only read below your classification and write above your classification

Mandatory Access Control: protect even against software/users (vs. Discretionary Access Control)
Picture: Confidentiality

Read below, write above

Proprietary

Public

S

Read above, write below

Proprietary

Public

S
Biba Integrity Model

- Rules that preserve integrity of information (dual to confidentiality)

- Two properties
  - No write up
    - A subject S may write object O only if \( C(S) \geq C(O) \)
      (Only trust S to modify O if S has higher rank …)
  - No read down
    - A subject S with read access to O may write object P only if \( C(O) \geq C(P) \)
      (Only move info from O to P if O is more trusted than P)

- You may only write below your classification and read above your classification
Picture: Integrity

Read above, write below

Read below, write above

Proprietary

Public

Proprietary

Public
Problems

- Covert channels
- Declassification
- Composability (e.g., one-time-pad)
- Overclassification (label creep)
- Aggregation
- Incompatibilities (e.g., upgraded files “disappear”)
- Polyinstantiation (“cover stories” and consistency)
Other policy concepts

**Separation of duty**
- If amount is over $10,000, check is only valid if signed by two authorized people.
- Two people must be *different*.
- Policy involves role membership and $\neq$.

**Chinese Wall Policy**
- Lawyers L1, L2 in same firm.
- If company C1 sues C2,
  - L1 and L2 can each work for either C1 or C2.
  - No lawyer can work for opposite sides in any case.
- Permission depends on use of other permissions.

These policies cannot be represented using access matrix.
Access control in web browsers

**Operating system**
- **Primitives**
  - System calls
  - Processes
  - Disk
- **Principals: Users**
  - Discretionary access control
- **Vulnerabilities**
  - Buffer overflow
  - Root exploit

**Web browser**
- **Primitives**
  - Document object model
  - Frames
  - Cookies / localStorage
- **Principals: “Origins”**
  - Mandatory access control
- **Vulnerabilities**
  - Cross-site scripting
  - Universal scripting
Components of browser security policy

Frame-Frame relationships

- canScript(A, B)
  - Can Frame A execute a script that manipulates arbitrary/nontrivial DOM elements of Frame B?

- canNavigate(A, B)
  - Can Frame A change the origin of content for Frame B?

Frame-principal relationships

- readCookie(A, S), writeCookie(A, S)
  - Can Frame A read/write cookies from site S?
Policies: further reading

- Anderson, Stajano, Lee, *Security Policies*  
  https://www.cl.cam.ac.uk/~rja14/Papers/security-policies.pdf

- Levy, *Capability-based Computer Systems*  
  http://www.cs.washington.edu/homes/levy/capabook
Information Flow Control

- An information flow policy describes authorized paths along which information can flow.
- For example, Bell-La Padula describes a lattice-based information flow policy.
Program-level IFC

- Input and output variables of program each have a (lattice-based) security classification $S()$ associated with them
- For each program statement, compiler verifies whether information flow is secure
- For example, $x = y + z$ is secure only if $S(x) \geq \text{lub}(S(y), S(z))$, where lub() is lowest upper bound
- Program is secure if each of its statements is secure
Implicit information flow

Conditional branches carry information about the condition:

```c
secret bool a;
public bool b;
...
// Incorrect:
b = a;
if (a) {
    b = 1;
} else {
    b = 0;
}
```

Possible solution: assign label to program counter and include it in every operation. (Too conservative!)
IFC Issues

- Label creep
  - Examples:
    - Branches
    - Logically-false flows
    - “Doesn’t really matter”
  - Branches

- Declassification
  - Example: encryption

- Catching all flows
  - Exceptions (math, null pointer, ...)
  - Signals
  - Covert channels
Noninterference

Experiment #1:

\[ p \]

\[ S_p = \{a\} \]

\[ p' \]

\[ S_{p'} = \{a\} \]

\[ \text{“HIGH”} \]

Experiment #2:

\[ q \]

\[ S_q = \{\} \]

\[ \text{“LOW”} \]
Implementation approaches

• Language-based IFC (JIF & others)
  – Compile-time tracking for Java, Haskell
  – Static analysis
  – Provable security
  – Label creep (false positives)

• Dynamic analysis
  – Language / bytecode / machine code
  – Tainting
  – False positives, false negatives
  – Performance
  – Hybrid solutions

• OS-based DIFC (Asbestos, HiStar, Flume)
  – Run-time tracking enforced by a trusted kernel
    • Works with any language, compiled or scripting, closed or open
Distributed Information Flow Control at the process level

See following lecture’s slides.