CSC 474/574 Information Systems Security

Topic 4.1: Basic Concepts of Access Control

Outline

- Access matrix model
- Access control lists versus Capabilities
- Content and context-based controls
- Discretionary versus mandatory controls
- Trojan Horses
- Bell-LaPadula model
## ACCESS MATRIX MODEL

### Basic Abstractions
- **Subjects**
- **Objects**
- **Rights**

The rights in a cell specify the access of the subject (row) to the object (column).
Users and Principals

- The system authenticates the user in context of a particular principal

![Diagram](image.png)
Users and Principals

- There should be a one-to-many mapping from users to principals
  - a user may have many principals, but
  - each principal is associated with an unique user
- This ensures accountability of a user's actions

In other words, shared accounts (principals) are bad for accountability
Principals and Subjects

- A subject is a program (application) executing on behalf of a principal
- A principal may at any time be idle, or have one or more subjects executing on its behalf
Principals and Subjects

- Usually (but not always)
  - each subject is associated with a unique principal
  - all subjects of a principal have identical rights (equal to the rights of the invoking principal)
- This case can be modeled by a one-to-one mapping between subjects and principals

For simplicity, a principal and subject can be treated as identical concepts. On the other hand, a user should always be viewed as multiple principals.

Objects

- An object is anything on which a subject can perform operations (mediated by rights)
- Usually objects are passive, for example:
  - File
  - Directory (or Folder)
  - Memory segment
- But, subjects can also be objects, with operations
  - kill
  - suspend
  - resume
ACCESS MATRIX MODEL

<table>
<thead>
<tr>
<th>Subjects</th>
<th>Objects (and Subjects)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

![Diagram of ACCESS MATRIX MODEL]

IMPLEMENTATION

- Access Control Lists
- Capabilities
- Relations
ACCESS CONTROL LISTS (ACLs)

- Each column of the access matrix is stored with the object corresponding to that column

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>r, w, own</td>
<td>r, w, own</td>
</tr>
<tr>
<td>G</td>
<td>r, w, own</td>
<td>r, w, own</td>
</tr>
</tbody>
</table>

CAPABILITY LISTS

- each row of the access matrix is stored with the subject corresponding to that row

<table>
<thead>
<tr>
<th></th>
<th>U</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F/r, F/w, F/own, G/r</td>
<td>G/r, G/w, G/own</td>
</tr>
</tbody>
</table>
ACCESS CONTROL TRIPLES

<table>
<thead>
<tr>
<th>Subject</th>
<th>Access</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>R</td>
<td>F</td>
</tr>
<tr>
<td>U</td>
<td>W</td>
<td>F</td>
</tr>
<tr>
<td>U</td>
<td>Own</td>
<td>F</td>
</tr>
<tr>
<td>U</td>
<td>R</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>R</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>W</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>Own</td>
<td>G</td>
</tr>
</tbody>
</table>

commonly used in relational database management systems

ACL'S VS CAPABILITIES

- ACLs require authentication of subjects
- Capabilities do not require authentication of subjects, but do require unforgeability and control of propagation of capabilities
ACL'S VS CAPABILITIES

• ACCESS REVIEW
  – ACL's provide for superior access review on a per-object basis
  – Capabilities provide for superior access review on a per-subject basis

• REVOCATION
  – ACL's provide for superior revocation facilities on a per-object basis
  – Capabilities provide for superior revocation facilities on a per-subject basis

ACL'S VS CAPABILITIES

• The per-object basis usually wins out so most Operating Systems protect files by means of ACL's

• Many Operating Systems use an abbreviated form of ACLs with just three entries
  – owner
  – group
  – other
ACL'S VS CAPABILITIES

• LEAST PRIVILEGE
  – Capabilities provide for finer grained least privilege control with respect to subjects, especially dynamic short-lived subjects created for specific tasks

CONTENT DEPENDENT CONTROLS

• content dependent controls such as
  – you can only see salaries less than 50K, or
  – you can only see salaries of employees who report to you
• are beyond the scope of Operating Systems and are provided by Database Management Systems
CONTEXT DEPENDENT CONTROLS

- context dependent controls such as
  - you cannot access classified information via a remote login
  - salary information can be updated only at year end
  - the company's earnings report is confidential until announced at the stockholders meeting
- can be partially provided by the Operating System and partially by the Database Management System
- more sophisticated context dependent controls such as based on past history of accesses definitely require Database support

DISCRETIONARY VERSUS MANDATORY

- Discretionary access controls (DAC)
  - Allow access rights to be propagated from one subject to another
  - Possession of an access right by a subject is sufficient to allow access to the object.
- Mandatory access controls (MAC)
  - Restrict the access of subjects to objects on the basis of security labels
    - Label both the subjects and the objects.
    - Allow a subject to access an object only when certain constraints are satisfied.
INHERENT WEAKNESS OF DAC

• Unrestricted DAC allows information from an object which can be read by a subject to be written to any other object
• Suppose our users are trusted not to do this deliberately. It is still possible for Trojan Horses to copy information from one object to another.

TROJAN HORSES

• A Trojan Horse is rogue software installed, perhaps unwittingly, by duly authorized users
• A Trojan Horse does what a user expects it to do, but in addition exploits the user's legitimate privileges to cause a security breach
TROJAN HORSE EXAMPLE

- Principal B can read contents of file F copied to file G

**Diagram:**
- Principal A performs actions on files.
- File G permissions: B:r, A:w.

**Procedures:**
- Trojan Horse executes
- Read: Program Goodies to File F
- Write: File G to Trojan Horse
- Principal B cannot read file F

**Consequences:**
- Principal B can read contents of file F copied to file G.
TROJAN HORSES

• Trojan Horses are the most insidious threat
• Viruses and logic bombs are examples of Trojan Horses
• It is possible to embed Trojan Horses in hardware and firmware
• It is possible to embed Trojan Horses in critical system software such as compilers and Database Management Systems

A Preview of A MAC Model

• Bell LaPadula (BLP) Model
  – Simple security: Subject S can read object O only if
    • Label(S) dominates label(O).
    • Information can flow from label(O) to label(S)
    • Intuitively, no read up
  – Star property: Subjects can write object O only if
    • Label(O) dominates label(S)
    • Information can flow from label(S) to label(O).
    • Intuitively, no write down.
BLP Model

Top secret
Secret
Confidential
Unclassified

dominance  Can-flow

Trojan Horse Example Again

User Alice executes Program Goodies

Trojan Horse

read Relation R

write Relation S

Alice: TS
Bob: S

Alice: r
Alice: w

Bob: r
Alice: w

The program cannot write to Relation S.