Computer Security
CS 426
Lecture 19

Discretionary Access Control
Access control

• Reference monitor mediate all access to resources
  – complete mediation: control **all** accesses to resources

[Diagram showing flow of access request through a reference monitor to a resource with a policy decision]

User process → Reference monitor (access request) → Resource

CS426 Fall 2010/Lecture 19
# ACCESS MATRIX MODEL

<table>
<thead>
<tr>
<th>Subjects</th>
<th>Objects (and Subjects)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>F</td>
</tr>
<tr>
<td>V</td>
<td>G</td>
</tr>
</tbody>
</table>

- **U** has **read own** rights to **F**.
- **G** has **read** rights to **U**.
- **V** has **read own** rights to **G**.

**rights**
IMPLEMENTATION OF AN ACCESS MATRIX

- Access Control Lists
- Capabilities
- Access control triples
ACCESS CONTROL LISTS (ACLs)

Each column of the access matrix is stored with the object corresponding to that column.
CAPABILITY LISTS

U: F/r, F/w, F/own, G/r

V: G/r, G/w, G/own

Each row of the access matrix is stored with the subject corresponding to that row.
**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 DBMS
ACL vs. Capabilities
The Confused Deputy Problem

System Admin

SYSX/FORT $OUTPUT Compiler Program

SYSX/BILL

SYSX (Dir)
FORT
STAT
BILL

Write to the bill file

Write output file

User

SYSX/BILL

$Output
Analysis of The Confused Deputy Problem

- The compiler runs with authority from two sources
  - the invoker
  - the system admin (who installed the compiler and controls billing and other info)
- It is the deputy of two masters
- There is no way to tell which master the deputy is serving when performing a write
How the Capability Approach Solves the Confused Deputy Problem

- Invoker must pass in a capability for $OUTPUT, which is stored in slot 3.
- Writing to output uses the capability in slot 3.
- Invoker cannot pass a capability it doesn’t have.
Capabilities vs. ACL: Ambient Authority

- Ambient authority means that a user’s authority is automatically exercised, without the need of being selected.
  - causes the confused deputy problem
- No Ambient Authority in capability systems
Capability vs. ACL: Naming

- ACL systems need a namespace for objects
- In capability systems, a capability can serve both to designate a resource and to provide authority.

- ACLs also need a namespace for subjects
  - as they need to refer to subjects
- Implications
  - the set of subjects cannot be too many or too dynamic
  - most ACL systems treat user accounts (principals) as subjects, and do not support fine-grained subjects
Capability vs. ACL: Authority Management

- Subject-Aggregated Authority Management
- In (almost) all ACL systems, the power to edit authorities is aggregated by resource
  - naturally compatible with Discretionary Access Control, where there is often the notion of an owner
- In capabilities systems, the power to edit authorities is aggregated by subject
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

- ACL's require authentication of subjects
- Capabilities do not require authentication of subjects, but do require unforgeability and control of propagation of capabilities
Conjectures on Why Most Real-world OS Use ACL, rather than Capabilities

- Capability is more suitable for process level sharing, but not user-level sharing
  - user-level sharing is what is really needed

- Processes are more tightly coupled in capability-based systems because the need to pass capabilities around
  - programming may be more difficult
Discretionary Access Control

- No precise definition. Basically, DAC allows access rights to be propagated at subject’s discretion
  - often has the notion of owner of an object
  - used in UNIX, Windows, etc.

"A means of restricting access to objects based on the identity and need-to-know of users and/or groups to which they belong. Controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (directly or indirectly) to any other subject."
Mandatory Access Control

- Mandatory access controls (MAC) restrict the access of subjects to objects based on a system-wide policy
  - denying users full control over the access to resources that they create. The system security policy (as set by the administrator) entirely determines the access rights granted
Multi-level Security (MLS)

- The capability of a computer system to carry information with different sensitivities (i.e. classified information at different security levels), permit simultaneous access by users with different security clearances and needs-to-know, and prevent users from obtaining access to information for which they lack authorization.
- Typically use MAC
- Primary Security Goal: Confidentiality
INHERENT WEAKNESS OF DAC

• Unrestricted DAC allows information from an object which can be read to any other object which can be written by a subject
  – do not provide multi-level security

• 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 cannot read file F

ACL

A:r
A:w

B:r
A:w

File F

File G
TROJAN HORSE EXAMPLE

Principal A executes Program Goodies

- **ACL**
  - A:r
  - A:w

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

- **Principal B**
  - B:r
  - A:w

- **Teddi: **
  - **ACL**
  - **Principal B**
  - B:r
  - A:w

- **File F**

- **File G**

- **Program Goodies**

- **Trojan Horse**

- **executes**

- **read**

- **write**
Buggy Software Can Become Trojan Horse

• When a buggy software is exploited, it execute the code/intention of the attacker, while using the privileges of the user who started it.
Readings for This Lecture

- Wikipedia
  - Discretionary Access Control
- The Confused Deputy by Norm Hardy
Coming Attractions …

- The Bell LaPadula Model