Protection

Chapter 14
Objectives

- Discuss the goals and principles of protection in a modern computer system
- Explain how protection domains combined with an access matrix are used to specify the resources a process may access
- Examine capability and language-based protection systems
Chapter Outline

- Goals and Principles of Protection
- Policy and Mechanism
- Domain of Protection
- Access Matrix
- Implementation of Access Matrix
- Access Control
- Revocation of Access Rights
- Capability-Based Systems (skip)
- Language-Based Protection (skip)
- Goals and Principles of Protection

- Goals of Protection
  - Operating system consists of a collection of objects, hardware or software
  - Each object has a unique name and can be accessed through a well-defined set of operations.
  - Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so.

- Guiding principle – principle of least privilege
  - Programs, users and systems should be given just enough privileges to perform their tasks
- Policy and Mechanism

- Good to separate protection policy from mechanism

- **Policy**
  - User dictates policy.
  - Who can access what object and in what mode.

- **Mechanism**
  - Operating system provides access-matrix + rules.
  - It ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced.
- Domain of Protection

- The ability to execute an operation on an object is an access-right

- Access-right = \( <\text{object-name}, \text{rights-set}> \)
  where \( \text{rights-set} \) is a subset of all valid operations that can be performed on the object.

- Domain = set of access-rights
System consists of 2 domains:
- User
- Supervisor

UNIX
- Domain = user-id
- Domain switch accomplished via file system (dynamic).
  - Each file has associated with it a domain bit (setuid bit).
  - When file is executed and setuid = on, then user-id is set to owner of the file being executed. When execution completes user-id is reset.
-- Example: Domain Implementation (MULTICS)

- Let $D_i$ and $D_j$ be any two domain rings.
- If $j < i \Rightarrow D_i \subseteq D_j$

![Diagram of rings](image)
- Access Matrix

- View protection as a matrix (access matrix)
- Rows represent domains
- Columns represent objects
- \( \text{Access}(i, j) \) is the set of operations that a process executing in Domain \( i \) can invoke on Object \( j \)
- Access matrix design separates mechanism from policy.
… - Access Matrix (static)

<table>
<thead>
<tr>
<th>domain</th>
<th>object</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>printer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td></td>
<td>read</td>
<td></td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>$D_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>print</td>
</tr>
<tr>
<td>$D_3$</td>
<td></td>
<td>read</td>
<td></td>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>$D_4$</td>
<td></td>
<td>read</td>
<td>write</td>
<td></td>
<td>read</td>
</tr>
</tbody>
</table>
Use of Access Matrix

- If a process in Domain $D_i$ tries to do “op” on object $O_j$, then “op” must be in the access matrix.

- Can be expanded to **dynamic** protection.
  - Operations to add, delete access rights.
  - Special access rights:
    - **owner** of $O_i$
    - **copy** op from $O_i$ to $O_j$
    - **control** – $D_i$ can modify $D_j$ access rights
    - **switch** - transfer from domain $D_i$ to $D_j$
### Access Matrix (Dynamic)

<table>
<thead>
<tr>
<th>domain</th>
<th>object</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>laser printer</th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td></td>
<td>read</td>
<td>read</td>
<td></td>
<td>switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_2$</td>
<td></td>
<td></td>
<td></td>
<td>print</td>
<td>switch</td>
<td>switch</td>
<td>switch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_3$</td>
<td></td>
<td>read</td>
<td>execute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_4$</td>
<td>read</td>
<td>write</td>
<td>read</td>
<td>write</td>
<td>switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
-- Access Matrix with *Copy* Rights

<table>
<thead>
<tr>
<th>Domain</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>execute</td>
<td></td>
<td>write*</td>
</tr>
<tr>
<td>$D_2$</td>
<td>execute</td>
<td>read*</td>
<td>execute</td>
</tr>
<tr>
<td>$D_3$</td>
<td>execute</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Domain</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>execute</td>
<td></td>
<td>write*</td>
</tr>
<tr>
<td>$D_2$</td>
<td>execute</td>
<td>read*</td>
<td>execute</td>
</tr>
<tr>
<td>$D_3$</td>
<td>execute</td>
<td>read</td>
<td></td>
</tr>
</tbody>
</table>

(b)
-- Access Matrix With *Owner* Rights

<table>
<thead>
<tr>
<th>object</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>owner</td>
<td>execute</td>
<td>write</td>
</tr>
<tr>
<td>$D_2$</td>
<td>read*</td>
<td>owner</td>
<td>read*</td>
</tr>
<tr>
<td>$D_3$</td>
<td>execute</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>object</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>owner</td>
<td>execute</td>
<td>write</td>
</tr>
<tr>
<td>$D_2$</td>
<td>owner</td>
<td>read*</td>
<td>read*</td>
</tr>
<tr>
<td>$D_3$</td>
<td>write</td>
<td></td>
<td>write</td>
</tr>
</tbody>
</table>

(b)
## Modified Access Matrix

<table>
<thead>
<tr>
<th>domain</th>
<th>object</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>laser printer</th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td></td>
<td>read</td>
<td>read</td>
<td></td>
<td>switch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_2$</td>
<td></td>
<td></td>
<td></td>
<td>print</td>
<td>switch</td>
<td></td>
<td></td>
<td>switch</td>
</tr>
<tr>
<td>$D_3$</td>
<td></td>
<td>read</td>
<td>execute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>switch</td>
</tr>
<tr>
<td>$D_4$</td>
<td></td>
<td>write</td>
<td>write</td>
<td></td>
<td>switch</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Access Control

- Protection can be applied to non-file resources

- Solaris 10 provides **role-based access control** to implement least privilege
  - Privilege is right to execute system call or use an option within a system call
  - Can be assigned to processes
  - Users assigned roles granting access to privileges and programs
-- Role-based Access Control in Solaris 10

user 1
role 1
 privileges 1
 privileges 2

executes with role 1 privileges

process
- Implementation of Access Matrix

- **Global Table**: Simple but usually too big to be kept in memory and difficult to take advantage of special grouping of objects or domains.

- Each column = **Access-control list** for one object
  Defines who can perform what operation.

  Domain 1 = Read, Write
  Domain 2 = Read
  Domain 3 = Read
  ...

- Each Row = **Capability List** (like a key)
  Fore each domain, what operations allowed on what objects.

  Object 1 = Read
  Object 4 = Read, Write, Execute
  Object 5 = Read, Write, Delete, Copy
- Revocation of Access Rights

- **Access List** – Delete access rights from access list.
  - Simple
  - Immediate

- **Capability List** – Scheme required to locate capability in the system before capability can be revoked.
  - Reacquisition
  - Back-pointers
  - Indirection
  - Keys
End of Chapter 14