Protection

Butler W. Lampson

Weina Ge & Divya Muthukumaran
Outline

- **Introduction**
  - What is protection
  - What are involved in the protection system
  - Why protection

- **Two abstract models**
  - Message system
  - Object system

- **Implement Issues & Comparisons between different mechanisms**
Protection

- Definition
  - All the mechanisms that control the access of a program to other things in the system

- Categories of protection mechanisms
  - Supervisor/user mode
  - Memory relocation and bounds register
  - Access control to file directories
  - Password logon
  - ...
Protection (Cont.)

- Different systems can adopt different protection mechanisms

- A single system can also have various methods to protect targets that of different characteristics
  - low-level hardware protection: TCP
  - high-level user interaction: password
Protection (Cont.)

- Why protection?
  - friendly and infallible users?
  - the real world is *dangerous*!

---

**single user**

**single process**

**single user**

**multi-process**

---

**multi-user**

**multi-processes**

DON’T open the door!
Motivation

- Keep the harm from spreading
  - Categories of infliction
    - Destroy of modifying others data
    - Accessing (r/w) without permission
    - Degrading the service another user gets (storage, CPU time)

- Different protection environments for contexts
  - protection context, environment, state, capability list, ring, *domain*
Message system

- A primitive protection system
  - Processes
  - Two characteristics:
    - Complete isolation - each process is a single domain
    - Sharing via message passing

- Message
  - Identification of the sender
    - Cannot be forged – supplied by the system
  - Data
Subroutine mechanism

- **Scenario**
  - Parameter

- **Protect the ‘entry’**
  - B: the supervisor & A: the user
    - B determines where to wait for A’s message

- **Protect the ‘return’**
  - B returns extra messages
    - Ignored because A knows exactly when to expect the returned message
  - B never returns
    - Check t1, t2

- Reliable

- t1

- t2
Disadvantages

- No control over a runaway process
  - Cannot force a process to do anything
  - Cannot destroy it

- Inconvenient to share information
  - have to share and check process names
SYSTEM MODEL

- USERS
- Protection domains: s1, s2, s3, s4, s5, s6
- Reference Monitor: policy
- Objects: o1, o2, o3, 04, 05, 06

Operations: d1, d2
ACCESS MATRIX

- A set of **OBJECTS** \((X)\) – processes, domains, files
  - That which is protected
  - Global validity (64-bit integer)
  - Object names handed out by the protection mechanism
- A set of **DOMAINS** \((D)\)
  - Entities having access to objects
  - Each domain has different access than other domains
- An **ACCESS MATRIX** \((A)\)
## Access Matrix

### Rows – Domain names

- **Domain 1**
- **Domain 2**
- **Domain 3**

### Columns – Object names

- **File 1**
- **File 2**
- **Process 1**

### Access Attributes – read, write

<table>
<thead>
<tr>
<th>Domains</th>
<th>Domain 1</th>
<th>Domain 2</th>
<th>Domain 3</th>
<th>File 1</th>
<th>File 2</th>
<th>Process 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain 1</td>
<td>*owner control</td>
<td>*owner control</td>
<td>*call</td>
<td>*owner</td>
<td>*read</td>
<td>*write</td>
</tr>
<tr>
<td>Domain 2</td>
<td>call</td>
<td>call</td>
<td>*read</td>
<td>write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain 3</td>
<td>owner control</td>
<td>read</td>
<td>owner</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Copy Flag Set
Implementation

Issues:
1) Memory protection is primarily provided by the hardware
2) Currently active/relevant parts need to be kept in cache
3) Grouping of objects/domains may be wasteful of resources

TYPES:
- Capability: Ticket-based
- Access Control Lists (ACL) : List-based
CAPABILITY

- Pairs $< x, A_{d,x} >$
- Associate $d^{th}$ column of $A$ with domain $d$
- Each domain is identified by a collection of capabilities called C-List

$< x_1, A_{d1,x1} >$
$< x_2, A_{d2,x2} >$
$< x_3, A_{d3,x3} >$
ACCESS CONTROL LISTS

- Attach protection information to the object
- Each object is associated with a list
  < domain name, access attributes >
- Group objects into directories
BANK ANALOGY (cont)

**LIST**

1) A list must be maintained for each vault
   Validity of the list must be ensured always

2) The bank must be able to verify the identity
   Of those wishing to access the box

3) To add a new person to a list the owner must verify himself and then have the new name added

4) A friend cannot extend his privilege to anyone

5) If a friend becomes hostile then the owner can just remove his name from the List

**KEY/CAPABILITY**

1) The bank need not be involved in the transaction once the keys are given

2) The lock and key system must be secure

3) The owner can just pass the key to anyone

4) Confinement problem

5) Key revocation
Memory Protection

- Each domain must have its own address space
- Paging
- Segmentation
References

- ACCESS CONTROL, Erik Poll, SOS Group, Radboud University
- The nucleus of a multiprogramming system, P.B. Hansen
- [http://zesty.ca/capmyths/](http://zesty.ca/capmyths/)
- Capability based computer systems, Henry Levy
THANK YOU!!!
Challenges in Access control

- **Assigning Permissions:**
  - Defining the Matrix
  - How to describe what we want?
  - Size and complexity

- **Checking Permissions:**
  - How to make sure that it is checked everywhere?