Advanced Authorization
and CORBA Access Control

Bob Blakley
IBM Lead Security Architect
April 1998
Authorization

subject security attributes

object control attributes

context attributes

Access Decision Function

access decision
Subject Security Attributes: Scale

<table>
<thead>
<tr>
<th>Thousands of users</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrea</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Messaoud</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tony</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Administrators can’t be expected to keep track of thousands of users, and know their authorization requirements, individually.
Subject Security Attributes Identities and Privileges

Subject: Bob Smith

- access_id: bobs
- group: dept573
- role: VP

Subject: Alice Jones

- access_id: a_jones
CORBA Authorization: Subject Security Attributes

Subject

Security Attributes

Access ID
Group(s)
Role(s)
Clearance(s)
Audit ID
....
CORBA Authorization: Passing subject security attributes to ADF

ADF Interface

Access Decision Function

subject security attributes

resource manager

request
Object Control Attributes

General form:

\[
\text{if } \{\text{condition}\} \\
\text{then} \\
[\text{subject}] \ [\text{may} \mid \text{may not}] \\
\text{do } [\text{action}] \text{ to } [\text{object}] \\
\]

Capability form:

\[
\text{if } \{\text{condition}\} \\
\text{then} \\
\text{this-subject } [\text{may} \mid \text{may not}] \\
\text{do } [\text{action}] \text{ to } [\text{object}] \\
\]

ACL form:

\[
\text{if } \{\text{condition}\} \\
\text{then} \\
[\text{subject}] \ [\text{may} \mid \text{may not}] \\
\text{do } [\text{action}] \text{ to this-object} \\
\]
Object Control Attributes: Scale

Millions of object instances

<table>
<thead>
<tr>
<th>obj_1</th>
<th>obj_2</th>
<th>...</th>
<th>obj_1000000000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Administrators can’t be expected to keep track of millions of object instances, and know their sensitivity, individually -- especially since many instances are created and destroyed dynamically and never even named.

An authorization system must provide a way of assigning the same policy to many different objects.
Object Control Attributes: Organization

Policy stored as unification rules

Problems

Difficulty of determining scope of rule
Rule interactions are very complex
All rules must be evaluated against all requests
Rule evaluation is slow
Object Control Attributes: Organization (2)

Policy stored in namespace entries

Issues:

Aliasing
Name Hierarchy is not sensitivity hierarchy
Object Control Attributes: Organization (3)

Policy stored as pattern-match rules

Problems

Aliasing
Difficulty of determining scope of rule
Rule interactions are very complex
Policy scope constrained by namespace
All rules must be evaluated against all requests
Object Control Attributes: Organization (4)

Policy associated with collections of objects
CORBA Authorization: Object Control Attributes

Object Instance    Domain Manager

Access Policy
Access Decision Functions

Arbitrary first-order predicate
First-order predicate without quantifiers
Attribute-value rule with equality only
Fixed-format rule with equality only
Subject-object-action rule
### Actions: Scale

<table>
<thead>
<tr>
<th>object(s)</th>
<th>class</th>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj_1, obj_2</td>
<td>Epson_printer</td>
<td>print, cancel_job, flush_queue</td>
</tr>
<tr>
<td>obj_8</td>
<td>HP_printer</td>
<td>submit_job, cancel_job, flush_queue, start_queue, stop_queue</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>obj_666</td>
<td>missile</td>
<td>arm, target, launch, abort</td>
</tr>
</tbody>
</table>

Administrators can’t possibly remember which class each of millions of objects belongs to.

Administrators can’t possibly remember what each method of each of thousands of classes does.
CORBA Authorization: Policy Language for Actions

The CORBA system defines a small, generic, extensible policy language, which developers and administrators use; admins don’t need to understand method semantics.
## CORBA Policy Management: Actions

<table>
<thead>
<tr>
<th>class</th>
<th>methods</th>
<th>required rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epson_printer</td>
<td>print, cancel_job, flush_queue</td>
<td>corba:use, corba:manage, set, corba:manage, set</td>
</tr>
<tr>
<td>HP_printer</td>
<td>submit_job, cancel_job, flush_queue, start_queue, stop_queue</td>
<td>corba:use, corba:manage, set, corba:manage, set, corba:manage, set</td>
</tr>
<tr>
<td>missile</td>
<td>arm, target, launch, abort</td>
<td>dod:missile_arm, dod:missile_target, dod:missile_launch, dod:missile_abort</td>
</tr>
</tbody>
</table>
CORBA Policy Management: Object Control Attributes (revisited)

Object Instance → Domain Manager → Access Policy

Maps Privilege Attributes to “Granted Rights”

Object Class → Required Rights

Maps Methods to “Required Rights”
Context Attributes

Ambient

  time/date

  location

  system state

Request-derived

  generic (traits common to all requests)

    quality of protection applied to request

    transited systems/applications

  specific (traits specific to a particular request)

    parameter value

Request/target derived

  caller is target owner

  caller is in role w.r.t. target (e.g. “primary provider”)
Context Attributes (2)

- Target
- Resource manager
- Environment
- Request
- ADF Interface
- Access Decision Function
- Request/Target derived attributes
- Request-derived attributes
- Ambient attributes
CORBA Authorization: Context Attribute Support

CORBA Authorization supports only ambient context attributes

ADF Interface

Access Decision Function

Environment

Subject security attributes

Ambient attributes

Resource manager

Request
CORBA Authorization: Policy Enforcement

access_allowed (obj_inst, operation, privattrs);
**CORBA Policy Enforcement Example**

**HP_Printer**

- **Required Rights**
  - flush_queue: corba: manage, set

**Public_Printers**

**Access Policy**

- bobs: use, set
- dept573: use
- a_jones: manage, set
- VPs: use

**Access Decision**

- access_allowed (901-laserjet, flush_queue, bob_sec_attrs);
- access denied

**Access Policy Example**

```corba
access_allowed (901-laserjet, flush_queue, bob_sec_attrs);
```
Encapsulation vs. Authorization Management

How does the security administrator know what objects Tony needs to be able to access?

How does Tony know which objects need to be able to use his identity in order to get his work done?
Compound Objects vs. Authorization Management

“Let Andrea update obj_1”

What does this command mean? Should Andrea be able to update obj_2, obj_3, and obj_4?

To which object is the command addressed? obj_1? Or a domain? Which domain? And how does a domain know what objects are contained in obj_1?

Who is authorized to issue this command?