Resource Access Decision Server: Design and Performance Considerations

Konstantin Beznosov and Luis Espinal
{beznosov,lespin03}@cs.fiu.edu

CADSE
October 22, November 5, 1999
Presentation Overview

• Introduction
• RAD Specification Overview
• RAD Prototype Design
• Performance Measurements
  – Model, Measurements, Results
  – Implementation Considerations
• Conclusions
Introduction: Access Control, etc.

• Access control
  – concerned with limiting activity of legitimate users
  – enforced by a reference monitor
• Authorization
  – concerned with making access control decisions
Access Control: Stand Alone vs. Distributed Systems

Stand Alone
- Primitive operations on objects controlled by OS (create, read, write, delete, use)
- Objects are homogenous (files, processes, memory)
- Single point of control
- Application access control is mangled with application logic

Distributed OO
- Stand alone systems, +
- Complex operations on interfaces
- Resources are heterogeneous (different interfaces),
- Many points of control (commonality, consistency, administration issues)
The Problem with Access Control in Distributed Systems

It is difficult to develop distributed systems that:

- insure commonality and consistency of policies
- perform security administration
- support access control for fine-grain resources
- allow changing policies without changing systems
- easy to verify and test
A Possible Solution

1. Application Request.
2. Authorization request.
3. Reply to authorization request.
4. Reply to application request.
Objective Statement

Study validity of the approach from the following perspectives

– Performance and scalability
– Ability to separate application logic from authorization logic (it works and performs)
– Ability to enforce complex policies and change them without pain
– Ability to test and verify application and authorization functionalities independently
Objective Analysis

• Why is this the right goal?
  – By solving it, we will be able to assess the validity of the approach
    • Help system designers and enterprise architects in constructing, verifying, and testing distributed systems.

• Why is the goal worth addressing?
  – It is doable
  – Its results could be applicable to other security policies and mechanisms (audit, quality of protection, non-repudiation)
Research Directions

+ Develop a prototype
+ Measure performance

• Study the validity of the main claims
  – support for different access control policy types
    • extend the prototype to support various policy types?
  – consistency and commonality of access control policies

• ???
RAD Specification

1. Application Request
2. Authorization Request
3. Reply to Authorization Request
4. Reply to Application Request
**RAD Specification: Component Collaboration**

1. `access_allowed(ResourceName, Operation, AttributeList)`
2. `get_policy_decision_evaluators(ResourceName)`
3. `get_dynamic_attributes(AttributeList, ResourceName, Operation)`
4. `combine_decisions(ResourceName, Operation, AttributeList, PolicyEvaluatorList)`
5. `evaluate(ResourceName, Operation, AttributeList)`
6. `* evaluate(ResourceName, Operation, AttributeList)`
Resource Access Decision Specification
Overview

---

**an Application System**

**an Access Decision Object : AccessDecision**

**a Locator : Policy EvaluatorLocator**

**an Attribute Service : DynamicAttributeService**

**a Combinator : DecisionCombinator**

**an Evaluator : PolicyEvaluator**

---

access_allowed(ResourceName, Operation, AttributeList)

get_policy_decision_evaluators(ResourceName)

get_dynamic_attributes(AttributeList, ResourceName, Operation)

combine_decisions(ResourceName, Operation, AttributeList, PolicyEvaluatorList)

* evaluate(ResourceName, Operation, AttributeList)
RAD Interfaces
Access Decision Object

<<IDL Interface>>
AccessDecision
(from ResourceAccessDecision)
- access_allowed()
- multiple_access_allowed()

<<IDL Interface>>
AccessDecisionAdmin
(from ResourceAccessDecision)
- get_policy_evaluator_locator()
- set_policy_evaluator_locator()
- get_dynamic_attribute_service()
- set_dynamic_attribute_service()

<<Interface>>
AccessDecisionExt

<<Interface>>
AccessDecisionExtOperations

<<Interface>>
AccessDecisionAdminExt
- shutdown()

<<Interface>>
AccessDecisionExtOperations

tie these two interfaces

+theAccessDecisionAdmin
Provides mechanisms to communicate with CORBA middleware

- **ComponentImplBase**
  - service()

- **Component**
  - service()

- **tieComponent**
  - delegate
    - tie.service() = delegate.serviceImplementation()

- **BOA**

- **ComponentOperationsImpl**
  - serviceImplementation()

- **ComponentOperations**

Tie Approach
Policy Evaluator Locator

<<IDL Interface>>
PolicyEvaluatorLocator
(from ResourceAccessDecision)

get_policy_decision_evaluators()

PolicyEvaluatorLocatorContext

set_default_evaluators()
get_default_combinator()
set_default_combinator()
get_default_evaluators()
get_policy_decision_evaluators()

<<IDL Interface>>
PolicyEvaluatorLocatorBasicAdmin
(from ResourceAccessDecision)

set_default_evaluators()
get_default_combinator()
set_default_combinator()
get_default_evaluators()

<<IDL Interface>>
PolicyEvaluatorLocatorAdminExt

tie
mechanism
Dynamic Attribute Service

<<IDL Interface>>
DynamicAttributeService
get_dynamic_attributes()

<<IDL Interface>>
DynamicAttributeServiceExt
shutdown()

<<IDL Interface>>
DynamicAttributeServiceExt
get_dynamic_attributes()

<<Interface>>
DynamicAttributeServiceExtOperations

<<Interface>>
DynamicAttributeServiceAdminExtOperations

tie mechanism

DynamicAttributeServiceContext

DynamicAttributeServiceStrategy
get_dynamic_attributes()

<<Interface>>
DynamicAttributeServiceContext

Strategy Pattern

EchoingDynamicAttributeService
get_dynamic_attributes()
Decision Combinator

- <<Interface>> DecisionCombinatorOperations
- <<IDL Interface>> DecisionCombinator combine_decisions()

- DecisionCombinatorContext
  - DecisionCombinatorContext() combine_decisions()
  - tie mechanism

- Strategy Pattern
  - 0..* -strategy 1..1

- Template Method Pattern
  - OpenWorldAndOrCombinationPolicy
    - grant access if no PE returns "NO"
  - ClosedWorldAndOrCombinationPolicy
    - grant access if all PE's return "YES"
Conducting Performance Measurements

- Measure response time perceived by the client: 
  \[ T_{emb} = (t_2 - t_1) \] and \[ T = (t_4 - t_3) \].
- Measure response time increase 
  \[ I = \left( \frac{T}{T_{emb}} - 1 \right) * 100 \]
- Repeat for 1ms, 10ms, 100ms, 1sec, 10sec business logic delays.
- Repeat for 1, 10, 100, 1000 authorization requests.
- Repeat for different configurations.
Test Configurations

Boundaries crossed: Application -> RAD/RAD Components
Host=ORB+network; Process=ORB+process; Object=function call
Conducting Performance Measurements

Network

- 24 Port 100Mb Switch
- 16 Port 100 Hub
- App Server Ext. Auth. Logic
- Client
- Naming Service
- PE
- RAD Server

conf. 3
conf. 5 & 6
conf. 4
Measurements Results

\[ I = \left( \frac{T}{T_{emb}} - 1 \right) \times 100 \]

<table>
<thead>
<tr>
<th>Application Processing Time/Authorization (ms)</th>
<th>Response Time Increase (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>69</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
</tr>
<tr>
<td>100</td>
<td>467</td>
</tr>
<tr>
<td>1000</td>
<td>466</td>
</tr>
<tr>
<td>10000</td>
<td></td>
</tr>
</tbody>
</table>

11/4/99
Factors affecting performance

• process co-location and direct (skipping middleware layers) invocations among RAD components
• host co-location of application and authorization servers
Conclusions

+ Prototype developed
+ Performance measurements collected
  • Preparing results for publication
  • Doing modeling of RAD and support for advanced access control policies