Access Control Architectures: COM+ vs. EJB

Dr. Konstantin Beznosov
Assistant Professor
University of British Columbia
Overall Presentation Goal

Learn about the capabilities of COM+ and EJB access control mechanisms
Konstantin
• Worked for end-user, consulting, and developer organizations
• Co-authored CORBA Security specification proposals
  - Resource Access Decision
  - Security Domain Membership Management (SDMM)
  - CORBA Security
• Co-authored
Conclusions

- Access control capabilities in both COM+ and EJB suck!
- One of them, however, is extensible
bullet Introduction
  - distributed security basics
  - distributed access control
  - evaluation criteria
bullet COM+
bullet EJB
bullet Conclusions
distributed security basics
requirements due to distribution

- centralized administration
- localized run-time decisions
object paradigm & security (1/2)

- **objects**
  - small amounts of data ==> large numbers
    - R: *Scale on large numbers of objects and methods*
  - diverse methods ==> complex semantics
    - R: *Security administrators should not have to understand semantics of methods*

- **collections**
  - R: *Similar names or locations should NOT impose membership in same collection(s).*
  - R: *For an object to be assigned to the same collection, name similarity and/or co-location should not be required.*
• many layers of indirection and late binding

• names
  - multi-name, nameless and transient objects
  - R: **Transient objects should be assigned to security policies without human intervention.**
  - less rigid naming hierarchies
  - R: **No assumptions that administrators know a name of each object in the system.**
distributed access control
Access Control at Run-time

Application

Target

Access Request

Enforcement Function

Decision

Middleware Security Subsystem

Decision Request

Decision Function

Middleware

Access Request
authorization decisions

- which policies?
  - which collections
- policy composition
- policy evaluation
  - information push vs. pull
evaluation criteria 1/2

- **GRANULARITY** -- granularity of protected resources
  - application, interface, method, arbitrary resource.

- **EXPRESSIVENESS** -- support for different access control models

- **RICHNESS** -- the variety of information available for making authorization decisions, including application-specific information

- **CONSISTENCY** -- support for consistency of policies across multiple applications
• MANAGEABILITY -- support for insertion and deletion of applications, changes in policies, user population and the computing environment
• SCALABILITY -- performance and administration scalability
• OBJECT PARADIGM REQUIREMENTS -- satisfying the requirements due to the object paradigm
• EXTENSIBILITY -- support for unforeseen policies
COM+
Administering Access Control
COM+ Access Control Architecture
scaling with collections

groups

permission collections via roles

components, interfaces

Clients

Methods

Subjects

Targets
• **Granularity**
  + component method
  – but not component instance method
• **Expressiveness** -- supporting different policies
  + \( \text{RBAC}_0 \)
  + \( \text{RBAC}_1 \) through W2K domain nested groups
• **Richness** -- information for making decisions
  - subject group attributes, component type and method
• **Consistency** -- across multiple applications
  - requires application redeployment, or manual changes in each application instance
evaluating COM+ 2/2

- **Manageability** -- changes to policies, users, appl-s
  - user population -- Windows domain groups could help
  - application population
    - replication -- easy to use packaging
      - new -- labor intensive and error prone
  - changes in policies -- labor intensive and error prone
  - computing environment -- easy to use packaging

- **Scalability** -- performance and admin. scalability
  - subject groups, several levels of permission granularity, permission collections
  - permissions (collections) local to the application

- **Object paradigm requirements**
  - roles isolate administrators from method semantics
  - machine co-located instances of the same component are governed by one policy

- **Extensibility** -- support for unforeseen policies
  - only through “programmatic security” inside of application
further reading on COM+


• MSDN Knowledge Base. http://msdn.microsoft.com
EJB
Defining Roles in EJB

<assembly-descriptor>
  <security-role>
    <description>
      blah-blah-blah …
    </description>
    <role-name>member</role-name>
  </security-role>

  <security-role>
    <description>
      blah-blah-blah …
    </description>
    <role-name>customer</role-name>
  </security-role>

  <security-role>
    <description>
      blah-blah-blah …
    </description>
    <role-name>staff</role-name>
  </security-role>

  ...
</assembly-descriptor>
Assigning Users to Roles in EJB

<security-role-mapping>
  <role-name>member</role-name>
  <principal-name>jgarcia</principal-name>
  <principal-name>mwebster</principal-name>
  <group-name>team-leads</group-name>
</security-role-mapping>

<security-role-mapping>
  <role-name>customer</role-name>
  <principal-name>dsmith</principal-name>
</security-role-mapping>
Assigning Methods to Roles in EJB

```xml
<method-permission>
  <role-name>staff</role-name>
  <method>
    <ejb-name>Product</ejb-name>
    <method-name>*</method-name>
  </method>
</method-permission>

<method-permission>
  <role-name>customer</role-name>
  <role-name>member</role-name>
  <method>
    <ejb-name>Product</ejb-name>
    <method-name>getPrice</method-name>
  </method>
</method-permission>
```
roles and permissions in EJB
scaling with collections

attributes

permission collections via method permissions

applications, beans

Clients

Operations

Subjects

Targets
Custom Authorization in EJB

- Java Authorization Contract for Containers (JACC) (formerly known as JSR 115)
  - Part of J2EE v1.4
  - defines an interface for pluggable authorization providers
Fine-grain authorization in EJB

isCallerInRole(role)
• **Granularity**
  + bean method in application
  - not at bean instance
  + arbitrary resource with security-role-ref

• **Expressiveness** -- supporting different policies
  + RBAC\(_0\)
    - RBAC\(_{1-3}\) -- product specific

• **Richness** -- information for making decisions
  - any user attributes are reduced to roles -- product specific

• **Consistency** -- across multiple applications
  - product specific
    - requires application redeployment, or manual changes in each application instance
evaluating EJB 2/2

- **Manageability** -- changes to policies, users, appl-s
  - user population -- product specific
    - application population
      + replication -- easy to use packaging
        o new -- labor intensive and error prone
    - changes in policies -- labor intensive and error prone
    + computing environment -- easy to use packaging

- **Scalability** -- performance and admin. scalability
  - subject groups -- product specific
  + three levels of permission granularity, permission collections
    - local to the application permissions (collections)

- **Object paradigm requirements**
  + roles isolate administrators from method semantics
  - container co-located instances of the same bean are governed by one policy

- **Extensibility** -- support for unforeseen policies
  - mostly through “programmatic security” inside of application
  + allows mapping from “external” to “internal” roles
  + JSR 115 “Java Authorization Contract for Containers” JACC
further reading on EJB


Conclusions

- Access control capabilities in both COM+ and EJB suck!
- EJB is extensible through JACC