A Comparison of Two Approaches for Achieving Flexible and Adaptive Security Middleware

Tom Goovaerts, Bart De Win, Wouter Joosen
DistriNet Research Group, KULeuven
Case Study
Personal Content Management
Evolution of Security Enforcement

Why?

- New legislation
- Changing business goals/partners
- New security requirements/threats
- Changes in environment (users, assets, ...)
- ...

Dynamic adaptations

- change a policy/security mechanism in response to an attack
- support dynamically negotiated security features
- ...

Dealing with Evolution

• Customizable middleware security
  • JAAS, JACC, IBM Tivoli Access Manager, RAD\textsuperscript{1}, Antigone Condition Framework\textsuperscript{2}, ...
  • No support for runtime adaptations

• Policy-based security enforcement
  • Trend towards declarative (authorization) policy languages (Ponder, XACML, EPAL, ...)
  • Dynamic changes supported by updating the policy
  • Many changes are related to policy deployment (configure the system so that it can evaluate a given policy)

1: K. Beznosov, Y. Deng, B. Blakley, C. Burt, and J. Barkley. A resource access decision service for CORBA-based distributed systems. ACSAC 99
2: P. McDaniel. On Context in Authorization Policy. SACMAT 03
General Research Topic

- Study of architectures for flexible & adaptive middleware security enforcement
  - Focus on enforcement of application-level authorization policies
    - Take application-level information into account in policy
    - Do not allow colleagues to view documents tagged ‘personal’
  - Security Service Bus
    - Support for late binding of security components via message bus
Focus of the Paper

- Integration of middleware security components & application components
  - Various mechanisms for composing middleware services with applications
    - Interceptors, reflection, AO, ...
  - Study how well these mechanisms support evolution in the context of security enforcement
Approach

- Security Service Bus-based integration
  - Composition of security services: contract-enhanced message bus
  - Integration with application components: middleware-level message interception

- AO middleware-based integration
  - AO middleware as an alternative technique to message interception + message bus

- Main contribution
  - Comparison of both approaches wrt support for evolution
Security Service Bus

- Framework for flexible middleware security services
  - Uniform policy domain model
  - Security component model
  - Security contracts specify dependencies
  - Message bus for late binding between security components
Policy Domain Model

- Policy-centric abstraction of environment (based on *)
  - OO Model based on Subjects, Actions, Resources + Attributes (-> XACML)
    - Which resources & attributes are globally present and usable in policies?
  - Independent from implementation of app components
  - Concepts do not need to map directly to implementation-level concepts

Security Components

Application Binding

Security Component

Application Components

Management
Information
Security

interception
Dependencies: Security Contracts

SecurityContract MetadataService {
  AttributeContract {
    provided {
      Document.owner;
      Document.sensitivity;
    }
  }
  AuthorizationContract {
    required {
      Document.read;
      Document.write;
    }
  }
}

SecurityContract PolicyEngine {
  AttributeContract {
    required {
      Document.owner;
      User.age;
    }
  }
  AuthorizationContract {
    provided {
      Document.read;
      Document.write;
    }
  }
}

Policy information

authorization requests
Security Service Bus

Application Bindings

obtain policy info

request policy decisions

manage

Security Administrator

Security Components

obtain policy info

request policy decisions

Policy Domain Model:
- Subjects
- Actions
- Resources

DistriNet
Composition

[Diagram showing a network architecture with various components and servers, including PeCMAN, Metadata, 3PS Binding, ACN Component, Secure Logger, and PDP.]
Adaptation Scenarios

- **S1** Change a local parameter of a security component
  - Trivial

- **S2** Introduce new security functionality
  - New security component (+ new class of contracts?)

- **S3** Compose/recompose a security component with one or more application components
  - Application Component-AB: costly, requires new code
  - AB-SC: domain membership

- **S4** Replace one instance of a security component with another one
  - Domain membership

- **S5** Compose a security component with a third-party component or service that is deployed elsewhere
  - New AB

- **S6** Change a security policy
  - Policy change = contract change, therefore unenforceable policy cannot be deployed
AO-based Composition

- AO Middleware
  - AO as programming model
  - AO to structure & customize middleware itself

- Features required for adaptations
  - Dynamic weaving
  - Programmatic runtime changes to aspect compositions
  - eg. JBoss AOP, Spring AOP
AO-based Composition

SomeAppComponent {
  getDocument(id) {...}
  addTag(id,tag) {...}
  removeTag(id,tag) {...}
}

ApplicationBinding {
  DocumentRead(id) {}  
  DocumentWrite(id) {}
  Object getDocumentOwner(id) {...}
}

AznComponent {
  checkPermission(s,a,r) {
    ...
  }
  ApplicationBinding ab = ...;
  Owner = ab.getDocumentOwner(r.id)
}

@Aspect("perthis(this(SomeAppComponent))")
BindingAspect {
  ApplicationBinding someappbinding=...;
  @Before(SomeAppComponent.getDocument)
  DocumentRead(id) {
    binding.DocumentRead(id)
  }
  @Before(SomeAppComponent.addTag &&
         SomeAppComponent.removeTag)
  DocumentWrite(documentid) {
    binding.DocumentWrite(id)
  }
}

@Aspect
AznAspect {
  AznComponent pdp;
  @Before(ApplicationBinding.DocumentRead)
  checkAccess(action,id) {
    // create s,a,r objects
    ...
    if(!pdp.isAuthorized(s,a,r))
      throw new SecurityException();
  }
}
Adaptation Scenarios

- S1 Change a local parameter of a security component
  - Trivial

- S2 Introduce new security functionality
  - New security component + aspect

- S3 Compose/recompose a security component with one or more application components
  - AB-SC: change pointcut of security aspect, hard to constrain/incorporate policy info
  - Application Component-AB: change pointcut of binding aspect

- S4 Replace one instance of a security component with another one
  - Change advice of security aspect

- S5 Compose a security component with a third-party component or service that is deployed elsewhere
  - Weave binding aspect in middleware-level proxy

- S6 Change a security policy
  - Composition does not take effective policies into account
Adaptation Scenarios

- S1 Change a local parameter of a security component
- S2 Introduce new security functionality
- S3 Compose/recompose a security component with one or more application components
- S4 Replace one instance of a security component with another one
- S5 Compose a security component with a third-party component or service that is deployed elsewhere
- S6 Change a security policy
Comparison

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSB</strong></td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td><strong>AO</strong></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>+</td>
</tr>
</tbody>
</table>

**DistriNet**

KATHOLIEKE UNIVERSITEIT
LEUVEN
DistriNet
Conclusions

1. AO more powerful alternative, given
   a) enough support for dynamic weaving and runtime modifications
   b) single homogeneous middleware platform

2. SSB takes dependencies based on policies into account
   + SSB provides maximum runtime flexibility
Future Work

- Taxonomy of security-related adaptations
- Practical comparison of both approaches on the same middleware platform + application
- Adaptation case study
  - Policy distribution