Security Engineering for Large Scale Distributed Applications

Konstantin Beznosov
Electrical and Computer Engineering
University of British Columbia
http://konstantin.beznosov.net
airplanes vs. cars

- flying is fast
- driving is slow
- why isn’t everybody flying?
why aren't secure systems everywhere?

almost completely insecure, or "secure" but

- too expensive and error-prone to build
- too complex to administer
- inadequate for real-world problems
- forever

examples
examples

- **CORBA Security**
  - no compliant system
  - over 600 pages
  - 3 days to install and configure a toy set up

- **Web services security**
  - harder than RPC-based CORBA
outline

- research direction
- access control mechanisms overview
- some things that can be done about it
- some specific things: attribute function, composable policy engines
- other research projects
what can be done about it?

- inexpensive and error-proof to build
- effective and inexpensive in administration
- adequate for problem domains
- easy and inexpensive to change and integrate
access control mechanisms overview
### Conventional Computer Security

#### Protection
- **Authorization**
- **Accountability**
- **Availability**
- **Access Control**
- **Data Protection**
- **Audit**
- **Non-Repudiation**

#### Assurance
- **Design Assurance**
- **Development Assurance**
- **Operational Assurance**

**Protection**
- preventing bad things from happening
- protection from breaking rules

**Assurance**
- enforces the rules, when rule check is possible
overview of access control

Specific to Application Domain

Request-specific

Subject Security Attributes
id=Alice  role=physician

Not managed by security admin-s

Context sensitive

Authorization Engine

Is attending physician?

Authorization Decisions

Reference monitor

Action Attributes
name=read

Access Control Mechanism

Subject Non-security Attributes
age=40

Object Attributes
patient=Bob  type=patient_record  subtype=current_episode

Object Security Attributes
owner=Fred  domain=hospital_A

“physician can read medical records”

“attending physician can modify patient current episode sensitive records”
decision-enforcement paradigm

- access control
- data protection
- security audit
some things that can make it better
separation of concerns

- application vendors – sell application(s) product
- middleware vendors – sell middleware products
- security vendors – sell security products
- application owners – sell service(s)
all security in middleware

Advantages
- “security-unaware” applications
- Both functions are implemented by security vendor
- Hard to bypass
- Separate from application – easier to analyze

Implementations
- CORBA, EJB, COM+, ASP.NET
- View Objects [Hailpern 1990]
- Role Classes [Barkley 1995]
- SafeBots [Filman 1996]
- Security Meta Objects [Riechmann 1998]
studying DF expressiveness

making better to administer

Security Policies

Policy Administration


Security Policy Mediators [Hale 1999]
Middleware security limitations

- no application-specific information or logic
- only information known before the method is invoked
- method-level granularity
reconfigurable decision function

Application space

Resource Access Decision (RAD)

Policy EvaluatorLocator

DynamicAttributeService

PolicyEvaluator

DecisionCombinator

Middleware Security

Application Object

Enforcement Function

Access Decision Object

1: Enforcement Function

2: get_policyDecision_evaluators

3: get_dynamic_attributes

4: combine_decisions

5: * evaluate


relationship-based access control

RBAC \otimes RAD == RelBAC

1: access_allowed

2: get_policy_decision_evaluators

3: get_dynamic_attributes

4: combine_decisions

5: evaluate

Application System

Access Decision Object

Policy Evaluator Locator

Policy Evaluator

Decision Combinator

Dynamic Attribute Service

RBAC

Relationships

RAD
one specific thing:
attribute function
enforcement in middleware

Application space

Advantages
- EF stays outside of the application
- DF can use application-specific policies and/or information

Examples
- CORBA Security replaceable mech.
- Java Authorization Contract for Containers, (JACC) in J2EE v1.4
- getAccess, Access Manager, SiteMinder
- Legion [Grimshaw 1998]

Middleware Security

Middleware Space

Application

Object

Decision Function

Enforcement Function

Access Request
how to get application data for decisions?

Disadvantages

- Each business object has to implement the backdoor
- Could be inefficient on expensive to activate objects
- Weak in the face of denial of service attacks
a better way – Attribute Function

Advantages
+ security out
+ application data in
+ separation of concerns
  - EF – middleware vendor
  - DF – authorization vendor
  - AF – application owner

another specific thing: composable policy engines

Distributed app. developers/admins have limited choices:

1. **Pre-built policy engines with limited capabilities**
   - e.g., JAAS default policy file, COM+, EJB authorization
   - Limited support for non-trivial or application-specific policies

2. **Pre-built policy engines “one size fits all” generic**
   - e.g., CORBA
   - Unnecessary complex and expensive to use

3. **“plug-in” APIs for creating custom “do-it-yourself” engines**
   - e.g., CORBA Sec. Replaceable, JSR 115, SiteMinder and alike
   - Hard to do it right
premise

- **common policy elements**
  - e.g., authorizations based on roles, groups, location

- **differences in**
  - the **weight and composition**
    - e.g., location || ( role && group ) vs. role || ( location && group )

- **application-specific factors**
  - e.g., relations, certification, license
component framework for A&A policy engine
expected benefits

- wide range of supported policies
- “pay as you go” cost of supporting a policy
  - determined by required policy
    - not by policy engine complexity
  - incremental changes proportional to policy $\Delta$-s
    - addition/removal/re-composition of policy components
  - re-use of existing policy logic by developers/administrators
example 1

university course web service
university course web service policy

1. Anyone can lookup course descriptions.
2. All users should authenticate using HTTP-Basic.
3. Registration clerks can list students registered for the course and (un)register students.
4. The course instructor can list registered students as well as manage course content.
5. Registered for the course students can download assignments and course material, as well as submit assignments.
policy engine assembly for example 1

Legend
- Generic
- Prebuilt
- Custom

- `<create>`
- `Credential HTTP_BA`
- `HTTP BA Credential Retriever`
- `Course PolicyEvaluator`
- `Publicmethods PolicyEvaluator`
- `Permit Overrides Decision Combinator`
- `Permission`
- `PermissionFactory`
- `CourseId AttributeRetriever`
- `QualifiedClassName`
- `MethodName`
- `TargetAttribute CourseId Attribute`
example 2

human resources web service for an international organization
HR web service policy

1. Only users within the company’s intranet or those who access the service over SSL and have valid X.509 certificates issued by the company should access.

2. Anybody in the company can look up any employee and get essential information about her/him.

3. HR employees can modify contact information and review salary information of any employee from the same division.

4. HR managers can modify any information about the employees of the same division.
policy engine assembly for example 2

(AuthorizedIP ∨ Certificate) ∧ (PublicMethod ∨ (Role ∧ Division))
Decision Combinator

Legend
Generic
Prebuilt
Generic from
Third-party
Custom
summary

▪ what
  • adequate for different application domains
  • inexpensive and error-proof to build
  • effective and inexpensive in administration and management
  • easy and inexpensive to change, and replace

▪ how
  • RBAC in CORBA
  • XACML
  • Resource Access Decision (RAD)
  • RelBAC
  • attribute function
  • composable policy engines
other research projects
multiple-channel SSL

- end-to-end security with partially trusted proxies
- selective data protection
usability of security administration

- improving visualization of the information
  - existing cognitive models of security administration

- improving feedback to security administrators
  - "what if" scenarios
  - safe staging playgrounds
  - testing system state

- better cognitive models

- mappings between different mental models/abstractions
  - application-specific model oriented on business processes
  - mechanism-specific technical model
1. examined the mismatch between security assurance and agile methods

2. classified conventional security assurance practices according to the degree of clash
   - 1) natural match, 2) methodology neutral, 3) (semi-)atomatable, 4) complete mismatch

3. suggested ways of alleviating the conflict
   - tool support, knowledge codification, agile-friendly assurance, intermittent assurance
Advanced ADAE/ADME Scheme

Client Tier
- Application Client
- Browser

Presentation Tier
- Web Servers

Component Tier
- Web Services
- Application Servers

Back-Office Tier
- Mainframes
- Databases

Application Infrastructure

EASI Application Environment Adapters
- Authentication API
- Authorization API
- Audit API
- Cryptography API

EASI Security Unifier

EASI Executive

Security Management
- Security Administration
- Security Configuration
- Security Policy

EASI Security Service Mappers
- Authentication Services
- Authorization Services
- Cryptography Services
- Accountability Services
- Security Administration Services

Component Tier
- Web Services

Presentation Tier
- Web Servers

Client Tier
- Application Client
- Browser
security diffuses in applications
attribute function in CORBA

Application space

Application

Attribute Function

Attribute Manager

Attribute Retriever

Enforcement Function
(security interceptor)

Decision Request

Decision Function
(DecisionObject)

CORBA Security

ORB

Security Domain