J2EE Security Model

Dan Johnsson
Omegapoint AB / Frobozz AB; Sweden
dan.johnsson@omegapoint.se / dan.johnsson@frobozz.se
Silly Example - Bar

Behind Counter

Bar

Who’s there?
- Women
- Men
- Personnel
- Security Guards

Ladies
Gents
Bar Activities

- Move around
  - Bar area
  - Behind counter
  - Ladies
  - Gents
  - Home

- Buy Drinks

- Take money from cash register

Who may do what?
The Story

- Use of Enterprise System

```
User DB
id cred
id cred
id cred

app/web
EJB

m RRR
m RRR
m RRR

ejb-jar.xml
```
J2EE

- Platform for enterprise systems
- Why do we need security model
Enterprise Systems

- Information system
  - Information = data about something
  - Manage data
    - Oppose: games, FEM-calculations
  - Producers and consumers
  - Consistency of data
  - Enterprise Logic = rules on data
  - System is Model of Reality
Large Scale Distributed Enterprise Systems

- Large Scale
  - Lots of users
  - Lots of kinds of users
  - Lots of functionality
  - Lots of kinds of use per user

- Distributed
  - No control of clients
Security Challenges

- Different access to functionality
  - Event within same component
  - *e.g.* register sale - sale statistics
  - Need authorisation system

- Functional components exposed
  - Risk for malicious clients
  - Need authentication
What Is in a Platform?

- Services provided by runtime environment
  - *e.g.*: memory management
    - **C++**: keep track of your object lest memory will leak
      - Learn strategies (object ownership etc)
      - Repetitive coding
    - **Java**: Garbage Collect
  - Like it or leave it
J2EE Platform Services (a few)

- Web tier: session tracking
  - HttpRequest.getSession()
- EJB tier: persistence service
  - EJB CMP
- Across all tiers
  - Standard Transaction Model
  - Standard Security Model
- All services: use or leave
Sound Use of Services

- 1 - Use (Standard Model)
  - Configure
    - majority of cases

- 2 - Augment
  - Build code on top
    - some cases

- 3 - Discard
  - Build your own system
    - very unusual
The Cube - Tiers

- Split system functionality
  - Client
  - Presentation
  - Enterprise/Business Logic
  - Integration
  - Resource

[Diagram of the Cube with tiers labeled: Client, Presentation, Enterprise/Business Logic, Integration, Resource. Binary code is included in the diagram.]
The Cube - Layers

- Layers of abstraction

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Platform</td>
</tr>
<tr>
<td>Operational Environment</td>
</tr>
<tr>
<td>Operating System</td>
</tr>
</tbody>
</table>

- Functionality
  - Buy - platform
  - Write - application
The Cube - Capabilities

- **Emergent properties**
  - Performance
  - Availability
  - Security
  - Extensibility

- Not located in single component
- Discussing architecture - must consider entire system
BarDweller - EJB Session

Session EJBs encapsulate enterprise flow

public interface BarDweller extends EJBObject {
    public void buyDrink() throws RemoteException;
    public void takeMoney() throws RemoteException;
    public void goBar() throws RemoteException;
    public void goBehindCounter() throws RemoteException;
    public void goLadies() throws RemoteException;
    public void goGents() throws RemoteException;
    public void goHome() throws RemoteException;
}

Dan Johnsson — J2EE Security Model
Security Threats (Examples)

- Crackers
- Denial of Service
- Burglary
- Blackmailing
- Inside jobs
Two Sides of Security

- Negative Security
  - What people must not be able to do
  - Risk: crackers

- Positive Security
  - What people must be able to do
  - Risk: denial of service
Security Policy

- Line between positive and negative
- Who is allowed to do what?
- Metric for measuring security
  - Can make automated tests
- Remember: System is Model of Reality
Bar Security Rules

- Only personnel/guards behind the counter
- Only women in the ladies (except guards)
  - Personnel in ladies must be women
- Only men in the gents (except guards)
  - Personnel in gents must be men
- Only personnel may handle money
- No drinking by personnel and guards
BarDweller - Security

- public void buyDrink()
  - all men and women, but neither guards nor personnel
- public void takeMoney()
  - personnel
- public void goBar()
  - everybody
- public void goBehindCounter()
  - personnel and guards
- public void goLadies()
  - women and guards
- public void goGents()
  - men and guards
- public void goHome()
  - everybody
Standard Security Model - Ambition

- Ambition: nice general level
- Considerations
  - Transparent for components
  - Declarative configuration
  - Extensible - possible to augment
    - Programmatic support
Standard Security Model - Design Fundamentals

- Method based access control
  - J2EE recommends Service Oriented Architectures
    - functionality grouped by themes
  - SOA demands method granularity
    - Functionality for mixed clients in one component
    - Component based access control - would not work

- Role based
  - Suffice for lots of applications
  - Still simple
Pipe Metaphor

authentication

app/web

authorisation

EJB

ejb-jar.xml

resource access

User DB

id cred
id cred
id cred

id cred
id cred
id cred

id RRR
id RRR
id RRR

id RRR
id RRR
id RRR

m RRR
m RRR
m RRR
Authentication

- Establish identity
- As close as possible to client
- Client container
  - Application container
  - Web container
- Transparent to code/web-app
- Client container holds identity
  - Pass in call to EJB-methods
User Database

- Keeps the user information
  - principal / credentials
- Operating environment, not covered by spec
  - Rigged by deployer / sysadmin
- Used by client container to verify identity
- Technology
  - Relational Database
  - LDAP Directory (preferred)

User DB

<table>
<thead>
<tr>
<th>id cred</th>
<th>id RRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>id cred</td>
<td>id RRR</td>
</tr>
<tr>
<td>id cred</td>
<td>id RRR</td>
</tr>
</tbody>
</table>
Application clients (J2EE [1.3] 9.2)

- Interact with user through JAAS
  - javax.security.auth.callback.CallbackHandler

- Configurable in deployment descriptor

- Flexible: Some possible authenticators
  - Pop-up window
  - Operating System
  - Biometric device
  - Smart Card
To Make It Work - GUI Client

- META-INF/application-client.xml

```xml
<application-client>
  <callback-handler>
    bar.auth.RetinaScanCallbackHandler
  </callback-handler>
</application-client>
```
Web Clients

- Web Container asks for proof of identity
  - BASIC, (DIGEST), FORM, CLIENT-CERT
- Identity stored in container (e.g. session)
- Identity passed to EJB on call
- Fixed identity: run-as (deploy setting)
  - Web container uses same ID
- Servlet [2.3] 12.5 (Security/Authentication)
To Make It Work - Web Login

- web-inf/web.xml
- `<web-app>`

  `<security-constraint>...</security-constraint>`
  `<login-config>`
    `<auth-method>BASIC</auth-method>`
  `</login-config>`
`</web-app>`
BASIC Authentication

- HTTP Authentication (RFC 2617)
- Username / Password Headers

HTTP GET

401 Unauthorized

Pop-Up

HTTP GET + Headers

User DB

id cred
id RRR
id cred
id RRR
id cred
id RRR

username/password?
Evaluation of BASIC

Advantages
- Simple

Drawbacks
- Credentials in plain text
  - HTTPS - all pages
    - Strain on web server
- Art Director
  - That pop-up looks ugly!
- Browser enforce security
  - Or so it seems to user
- Browser keeps credentials
  - Bad for public terminals
  - No "clear credentials" button
FORM-based Authentication

- Challenge - Response
- Challenge = login form
  - j_username, j_password, j_security_check

HTTP GET
redirect to form
<form ...>
HTTP POST

User DB
id cred
id cred
id cred
id RRR
id RRR
id RRR

username/password?
Evaluation of FORM

- **Advantages**
  - Control over Look-and-Feel of login form
  - Can secure just login
    - Redirect to HTTPS-URL

- **Drawbacks**
  - Hard to add extra info to login form
    - *e.g.* “login failed”
  - Credentials saved by some browsers
CLIENT-CERT Authentication

- Client present certificate w public key
- Server open SSL-tunnel
- Data encryption not really authentication
  ➢ challenge-response auth during handshake
Evaluation of CLIENT-CERT

- Advantages
  - Very secure

- Drawbacks
  - Demands PKI
    - Cumbersome admin
The Story This Far

Diagram:
- A stick figure
- App/web: ID
- EJB
- User DB: id cred
- ejb-jar.xml

Legend:
- O: Object
- ID: Identifier
- EJB: Enterprise Java Bean
- db: Database
- ejb-jar.xml: Enterprise Java Bean archive XML file
Call to EJB

- Identity passed in call
  - Packed in network package
    - EJB [2.0] 19.8.2 Securing EJB invocations

```
Call to EJB

- Identity passed in call
  - Packed in network package
    - EJB [2.0] 19.8.2 Securing EJB invocations
```
Authorisation

- By EJB container
- Transparent
- Fetch ID from call
- Fetch roles from user database
- Fetch permitted roles from EJB DD
- Role resolution
Role Resolution

- Automatic by the platform
- Programmatic
  - EJBCoNtext.isCallerInRole
  - EJBCoNtext.getCallerPrincipal
To Make It Work - Roles

- Application
  - Define application roles
  - Assign access permissions
- ejb-jar.xml
  - `<assembly-descriptor>`
    - `<security-role>`...
    - `<security-role>`...
    - `<method-permission>`
      - `<role-name>`...
      - `<method>`...
      - `<method>`...
    - `</method-permission>`
  - `</assembly-descriptor>`

- Operating Environment
  - Map principals to application roles
  - Managed through app-server tool
BarDweller - Security

- public void buyDrink()
  - all men and women, but neither guards nor personnel
- public void takeMoney()
  - personnel
- public void goBar()
  - everybody
- public void goBehindCounter()
  - personnel and guards
- public void goLadies()
  - women and guards
- public void goGents()
  - men and guards
- public void goHome()
  - everybody

Hmm, tricky
Let’s ignore
To Make It Work - ejb-jar.xml

- Security Roles

  - `<security-role>`
    - `<description>`
      - male or ...
    - `<role-name>`man`<role-name>`
  - `<security-role>`
    - `<role-name>`guard`<role-name>`
  - `<security-role>`
    - `<role-name>`woman`<role-name>`

- Method permissions

  - `<method-permission>`
    - `<role-name>`woman`<role-name>`
    - `<method>`
      - `<ejb-name>`BarDwellerEJBean`<ejb-name>`
      - `<method-intf>`Remote`<method-intf>`
      - `<method-name>`buyDrink`<method-name>`
    - `<method>`
      - ... goLadies`<method>`
  - `<method-permission>`
The Story This Far

User DB

id cred id cred id cred
id cred id cred id cred
id cred id cred id cred
Resource Control

- Containment
  - Limits access to resource
- Exposure
  - Gives controlled access to resource
Example: Log Database

- **Problem**
  - Database w/ complex table relations
  - Do not want access code throughout system

- **Solution**
  - Database hidden behind EJB
  - EJB configured with credentials
  - Database inaccessible “from side”
Resource Authentication
Requirements (J2EE 1.3 spec)

- Required
  - Configured Identity
  - Programmatic Authentication

- Recommended
  - Caller Impersonation
  - Principal Mapping
  - Credential Mapping
Configured Identity

- **Standard model**
- **Username / password**
  - Held by EJB
  - Set by deployer
  - Static
- **Code:**
  - `datasource.getConnection()`
Programmatic Authentication

- If standard model not good enough
- Code it in component (app layer)
- Code:
  - `datasource.getConnection(user, passwd)`
- Completely general
Recommended Features

- Caller Impersonation
- Principal/credentials mapping
- Why are they recommended?
  - Handy in some usual cases
Example

- Legacy system
  - Database
  - GUI clients
  - Users in database

- Webbify it!
  - What ID should Web system use?
  - Need to use client’s ID
Caller Impersonation

- User caller ID to contact resource
Principal Mapping

- Principal is not always same
- Principal ‘danj’ to database
- Principal ‘dan.johnsson@omegapoint.se’ to Web system
Credential Mapping

- Not only different principals
- Different passwords
- Different ways of authentication
  - Certificate to web tier
  - Username/password to database
- Credential mapping translates
The Story This Far

![Diagram](image)

- User DB
  - id cred
  - id cred
  - id cred

- ejb-jar.xml
  - m RRR
  - m RRR
  - m RRR

- app/web
- EJB

Dan Johnsson — J2EE Security Model
Limitations of Model

- Complicated role systems
  - negative permissions
- Data-oriented security
  - Balance of bank account
    - Not for all “Web-user”s - just for me
public void buyDrink()
  ➢ all men and women, but neither guards nor personnel
public void takeMoney()
  ➢ personnel
public void goBar()
  ➢ everybody
public void goBehindCounter()
  ➢ personnel and guards
public void goLadies()
  ➢ women and guards
public void goGents()
  ➢ men and guards
public void goHome()
  ➢ everybody
Augmenting the Model

- Do not throw out model
  - baby / bath water

- Code on top of model

```java
public void buyDrink() {
    if(sessionctx.isCallerInRole("personnel")
        || sessionctx.isCallerInRole("guard"))
        throw new SecurityException("Not permitted to drink");

    ...
}
```

- Note: Seemingly hardcoded “personnel” and “guard” are mapped to application roles.
Data-oriented Security

- Example: Internet bank w/ account
- Role based security not enough
- Augment model: code on top

  ```java
  public void getBalance() {
      if(! areSame(getName(), ctx.getCallerPrincipal()))
          throw new SecurityException("Not your account");
...
  ```
Recap of Full Scale Model

- Authentication by client containers
- ID transport on call - transparently
- Authorization by EJB container
- Protected resources
- todo: What triggers authentication?
- todo: User Administration
Web System

- Degenerated case
- No EJB tier
  - Who will authorise?
- Web container have to - no other around
  - Protected views
- Will trigger authentication!

![Diagram showing user, web container, and user database with user identifiers and credentials]
To Make It Work

- Example web.xml (SRV[2.3] 13.4.2)

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>SalesInfo</web-resource-name>
    <url-pattern>/salesinfo/*</url-pattern>
    <http-method>GET</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>manager</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```
User Administration

- Large system $\Rightarrow$ dynamic user base
- Black Sheep - no standard
- “This scenario was widely discussed /.../ [JSR-53 EG] but we were unable to achieve consensus on the appropriate solution. We had to abandon this work for J2EE 1.3 ....”
Drawbacks in User Admin

- No User Admin API
- Cannot add users
- Cannot change password
- Cannot change roles
- Workarounds Exist!
User Databases

- Not standardised

- Usual
  - Flat file
  - Relational database
  - LDAP system

- Most app-servers
  - Proprietary interface
  - Adapters
User DB Example - Tomcat

- Interface
- Adapters
  - MemoryRealm
  - JDBCRealm
  - JNDIRealm
- Config in server.xml
Workaround for User Admin

- Use your LDAP system
  - good industry support
- Update through JNDI
- No portability guarantee
Crystal Ball

- User admin API
- Connectors to auth systems
Future JSRs

- Java Authentication Service Provider Interface for Containers (196)
- Java Authorization Contract for Containers (115)
Conclusions

- Standard model in platform
- Usable in most cases
- Declarative - transparent to code
- Can be augmented programmatically
- User admin is black sheep