Middleware security
J2EE

François Staes
NetConsult BVBA
fstaes@netconsult.be
Introduction

- What is J2EE (helicopter view)
- Traditional J2EE Security subjects:
  - Authentication
  - Authorization
  - J2SE Security
- Other J2EE Security subjects
  - Trust boundaries
  - Scalability versus Security
What is J2EE
J2EE

- Standardized middleware component architecture

- **Component architecture:**
  A *contract* between you (the component developer) and them (the application servers vendors)

- **Standardized:**
  Once your components obey the contract, they can be deployed in any J2EE-compliant application server
The J2EE architecture clearly distinguishes different roles:

- Application Component provider (**you**)
- Application Assembler
- Deployer
- System administrator
- Application Server provider (**them**)
J2EE component architecture

**your responsibilities**

- Focus on business logic
- Don’t care about infrastructural issues as:
  - Network communication protocols
  - Transactional integrity
  - Persistency
  - Security
  - Scalability
  - Failover
  - ...

J2EE component architecture

Their responsibilities

- Take care of all the infrastructural issues
- Differentiating factors:
  - Performance
  - Scalability
  - Failover
  - Additional features
    - not specified by the J2EE specs
    - proprietary by definition
J2EE
Types of Components

- **Web-centric**
  - Servlets
  - Java Server Pages (JSP)
  - Java Server Faces (JSF)

- **Enterprise JavaBeans**
  - Stateful / Stateless Session Beans
  - Message-driven Beans
  - Entity Beans

- **Java Connector Architecture**
  - Resource Adapters
J2EE Security

Traditional subjects
J2EE Security

Traditional Subjects

- Authentication:
  - Provided by the container vendor
  - Configured by the deployer

- Authorization
  - Access rules specified by developer
    - Mostly declaratively (config files)
    - Possible to do so in code too
  - Access rules checked for by the container
J2EE Authentication

Differs between access protocols:

- **HTTP:**
  - Basic Authentication
  - Client certificates
  - Form-based

- **WebServices:**
  - WS-Security support required by specs (basic, X.509)

- **CORBA:**
  - CSIv2
J2EE
Authentication in real-life

- Lots of proprietary extensions
  - E.g. WebSeal
- HTTP / SPNEGO
  - fully supported in Weblogic & WebSphere
  - supported by third-party plugin in JBoss
- SAML
  - Fully supported in Weblogic
  - Supported in the context of WebServices in WebSphere
- No standardized way to add support for new protocols
J2EE Authentication
The verification process

- How does the container verify your credentials (password)?
- Implemented by means of JAAS Logic Modules
  - Similar to Pluggable Authentication Modules (PAM) as known in the UNIX environments.
- Can make use of Callbacks when used in the context of application clients.
J2EE Authorization

- Developer specifies that a component is only accessible to role ‘Admin’.
- Done by means of configuration (deployment descriptors)
- Or programmatically:
  - `getCallerPrincipal()` / `getUserPrincipal()`
  - `isCallerInRole(String role)` / `isUserInRole(String role)`
J2EE Authorizations

- What does the role ‘Admin’ translate to in your environment?
- Remember you might buy third-party components
- Map ‘Admin’ to a number of users/groups/roles in your environment
J2EE Authorizations

➢ Declaratively:
  • The deployer maps the role ‘Admin’ to security identities known in your environment
    • Principals, groups, ...

➢ Programmatically (since J2EE 1.4):
  • Use JACC (Java Authorization Contract for Containers)
  • Plug-in modules to implement ‘isCallerInRole’ and friends
J2EE Security versus J2SE Security

- J2EE Authorizations: Which user has access to what functionality (web page, EJB method)
- J2SE Authorizations: Which codesource has access to what resource (file, network connection, ....)
- Is there any relationship between both???
J2SE Security
in the context of J2EE

- Protect the application server from malicious components
  - Realistic?
- Implement principle of least privilege
  - Does your servlet need to be able to call ‘System.exit(0)’?
- Typically not enabled by default
J2EE/J2SE Authorization Configuration

If your component needs certain privileges:

➢ Use AccessController.doPrivileged

```java
AccessController.doPrivileged(new PrivilegedAction() {
    public Object run() {
        //
        }
    }
});
```

➢ Add the authorization to:

• The deployment descriptor (weblogic)
• Global policy file (others)
J2EE Security

Other subjects
The first half of my talk discussed the more ‘development’ aspects of the J2EE security.

It explained the responsibilities of the different parties involved, supposing that every party performs its duty.
J2EE
Defining Trust Boundaries

- Can we trust all the parties involved?
- Define the boundaries of our trust
- Implement proper safety guards whenever interacting with a party we cannot fully trust
J2EE Trust Boundaries

Elements involved:

- Client applications
- Network between client and application server
- Application Server
- Components
- Network between application server nodes
  - Web-tier to business tier
  - Within tier (used in clustering)
- Database server
J2EE Trust Boundaries

Clients

- Never trust input validations performed by the client
- Never trust state information maintained by the client
J2EE Trust Boundaries
Client-Server network

- Only allow incoming traffic on the ports you expect it
  - Firewalls
  - Demilitarized zones

- Confidentiality / Integrity / Authentication
  - SSL

- Non-Repudiation
  - No standard solution
J2EE Trusted Boundaries
Application Server

- Not trusting your application servers is difficult
- Similar to not trusting your operating system
J2EE Trust Boundaries
Components

- Maybe not that important if your component is the only one deployed on the application server

- But what if your application is deployed on a shared application server?

- Solved by means of the J2SE security architecture
J2EE Trust Boundaries
Application Server interconnections

- **Inter-tier**
  - web tier talking to business tier

- **Intra-tier**
  - Nodes in one tier talking between them
  - Mostly to maintain global state information
J2EE Trust Boundaries
Inter-tier communication

- Typically involves crossing network segregation boundaries
- Mostly the same issues as for client-server communication
- Configure business tier to only allow incoming traffic from the web-tier servers
J2EE Trust Boundaries
Intra-tier communication

- Mostly used to exchange state information
  - When storing state information in HttpSession objects (web-tier)
  - When using stateful session beans (EJB)
  - Synchronizing entity beans

- Use private network

- Most application servers don’t allow one to use SSL for this kind of communication
J2EE Trust Boundaries
Database access

- Most infrastructures rely on a DB connection pool
- This implies that the DB doesn’t know the full identity of the end-user
  - Reasonable assumption for Internet applications
  - Often not so reasonable for Intranet applications
- Improves with JDBC 4.0 (future)
Scalability and/versus Security

- While properly defining trust boundaries allows us to deny unlawful access to our data, it does not protect us from denial-of-service attacks.

- Defeating them at the network level is one solution, but defeating distributed DOS attacks that way is difficult.
Scalability

- Creating a scalable solution is a first step to defeat DOS attacks.

- Scalability does not go well together with stateful
  
  - Either maintain the state information on a single node;
  
  - Either replicate the state information to other nodes
Scalability
Security issues

- The more nodes one introduces the more security risks one takes
- Use dedicated (isolated) network for intra-tier communication
- Is failfast a solution in this context?
  - How to detect corrupted nodes?
  - Defeats scalability, makes DDOS attacks really easy!
Replicating state information

- Naïve approach: replicate to all other nodes

- More advanced approaches:
  - Replicate to N other nodes
  - Replicate to 1 other node. If the primary or secondary node fails, choose another one
  - Replicate to a set of dedicated nodes
  - …
Questions ?