Java EE - Security

Ari M. Ayvazyan, Adrian W. Bergler

16.02.2015

Agenda

- **Introduction to Secure Systems & Architecture**
- Authentication & Authorization
- Deployment Descriptors
- **Live example**
- **Enterprise Beans**
- Application Client
- Digital Certificates
- Frameworks
- **Summary & Questions**

Secure Systems

- CIA-Triad
  - Confidentiality
  - Integrity
  - Availability
- Non-Repudiation

Confidentiality

- Data can‘t be read by unauthorized

Integrity

- State of system can only be „correctly“ modified

Availability

- How stable the system is against errors and attacks
- Being available in general

Non-Repudiation

- Actions only be executed by their defined identities (eg. user groups)
- Also: Ability to make users accountable for their actions

Introduction

Transport Layers	• Transport Security - VPN
Java EE	• Users, Logins, Permissions • Https
SQL	• SQL Users & Permissions
Java EE	• Output Escaping

Possible Security Implementations

- **Declarative Security** (this includes **@Annotations** and **XML-Files**)
  - applied by the **container** during **deployment**.
- **Programmatic Security**
  - applied by itself **at runtime**.

Authentication & Authorization

- **Authentication**
  - *Who are you?*
  - Identification
- **Authorization**
  - *What are you allowed to do?*
  - Assignment of Permissions to a ***Authenticated*** User

Deployment Descriptors

- Describes how the Application should be Deployed.
- Defines Security Constraints
  - Protected Information
  - Probably SSL
  - Specify which user may access them
- XML-Files
- Usually located in /WEB-INF/
  - web.xml
  - Vendor-specific.xml (E.g. Glassfish: glassfish-web.xml)

Deployment Descriptors **web.xml**

- Protected Resources
- Security Roles
- Authentication methods

Deployment Descriptors **(vendor-specific).xml**

- User – Role mapping
- Group – Role mapping

- Vendor specific settings of a **container**

Principals, Credential

- A **Principal** is a **identity** that can be authenticated.
  - E.g. a Unique Username
- A **Credential** is defined as information that is used to authenticate a Principal.
  - E.g. a Password

Groups, Roles

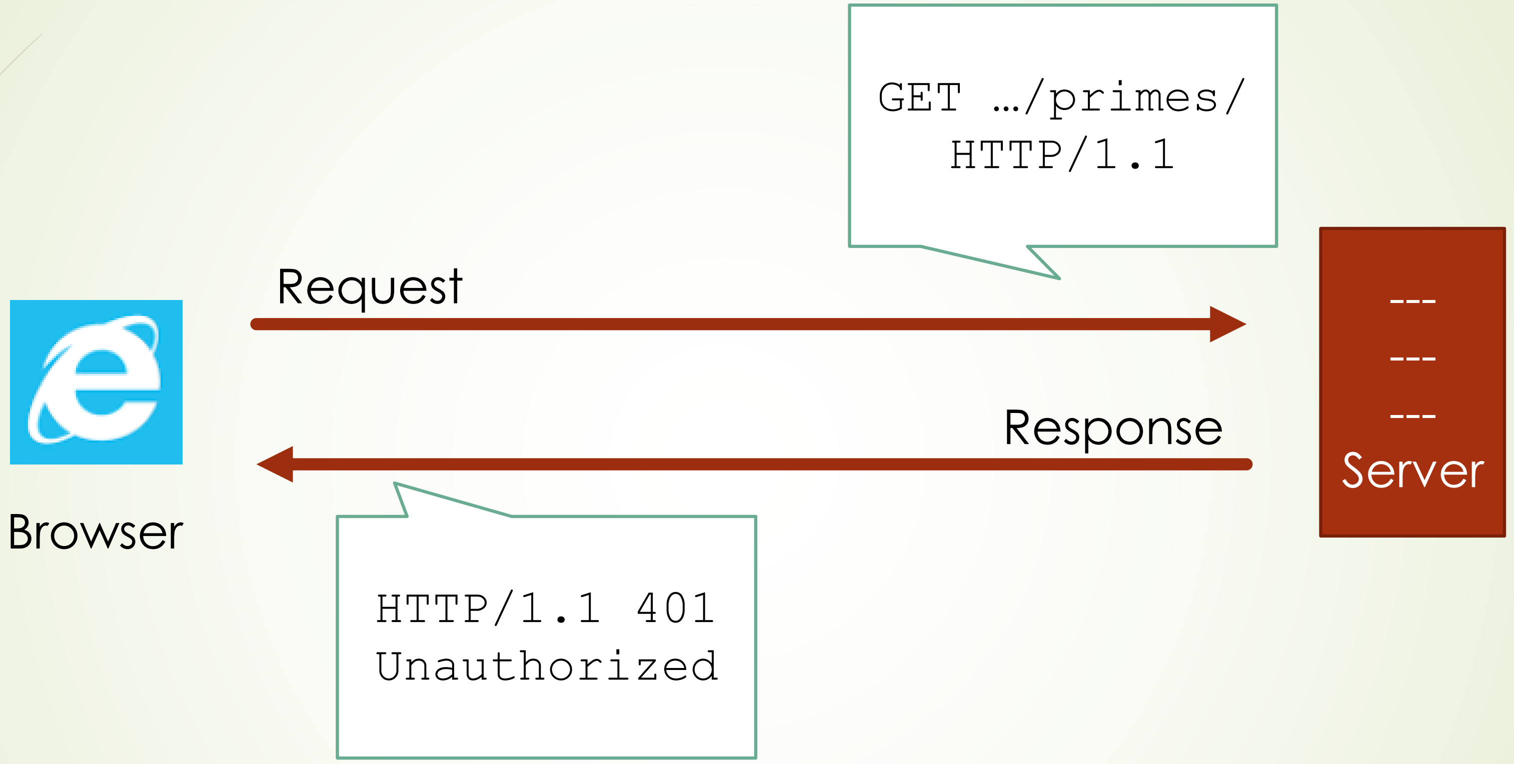
- **Permissions** are granted to **Roles**.
- **Groups** and **Principals** can be mapped to **Roles**.
- **Roles** are defined in web.xml
- **Groups** are defined in vendor-specific.xml

Realm

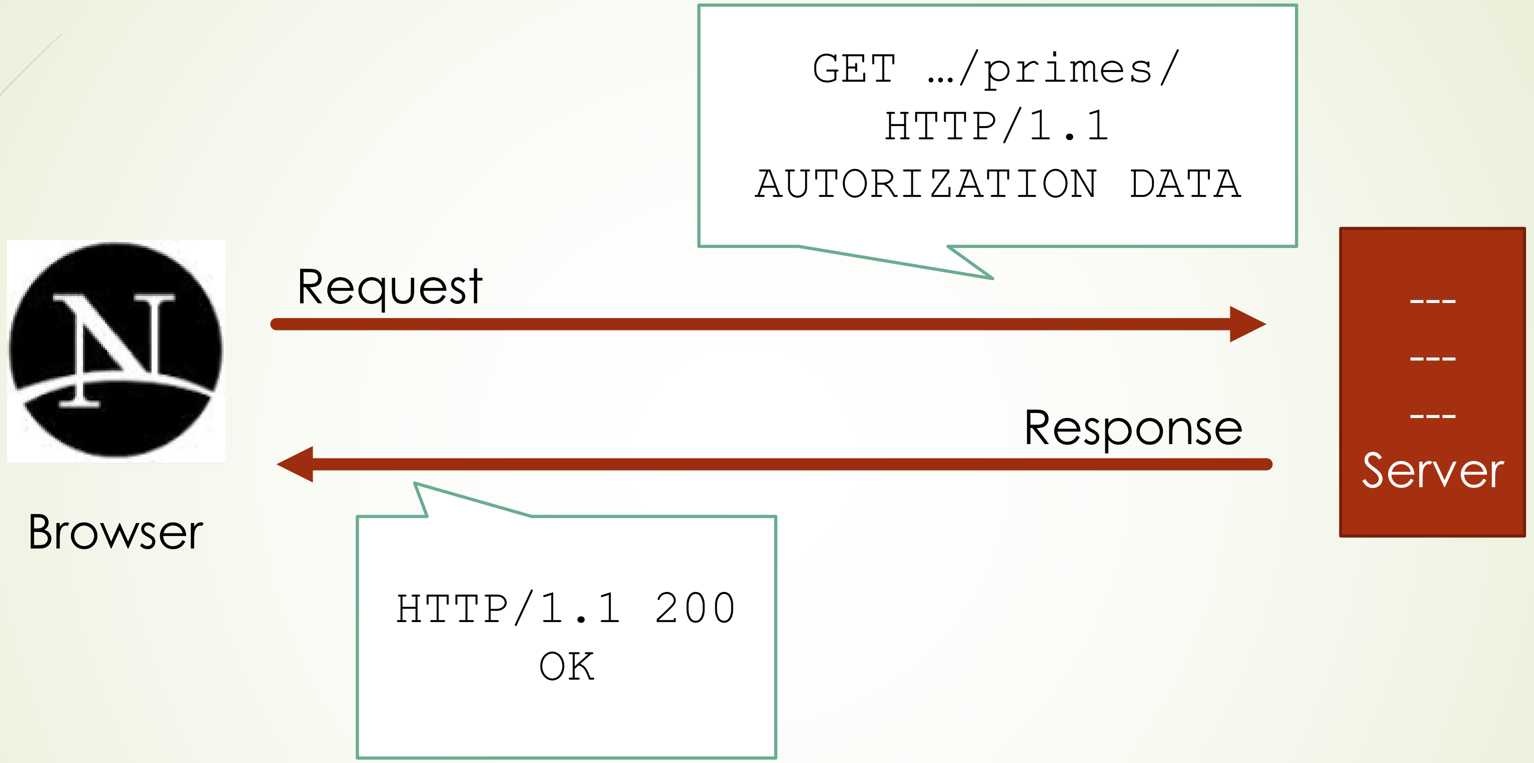
- Provides information about **principals**, their **Groups** and their **credentials**
- May be a Database, File structure, connection…

- **In other words**
  - Contains User Information
    - E.g. Username, Password & Permissions

Live example preview



Live example preview



Live example

https://github.com/aayvazyan-tgm/JavaEESecurityExample

Enterprise Java Beans

- **Role-based authentication**
- Limit access to certain roles
  - Beans (whole class)
  - Methods **independent** of their signature
  - Methods **with definition** of their signature

Enterprise Java Beans

- Declarative
  - Through **deployment descriptor**
- Annotations
- Programmatic
  - **Directly** in **source** code

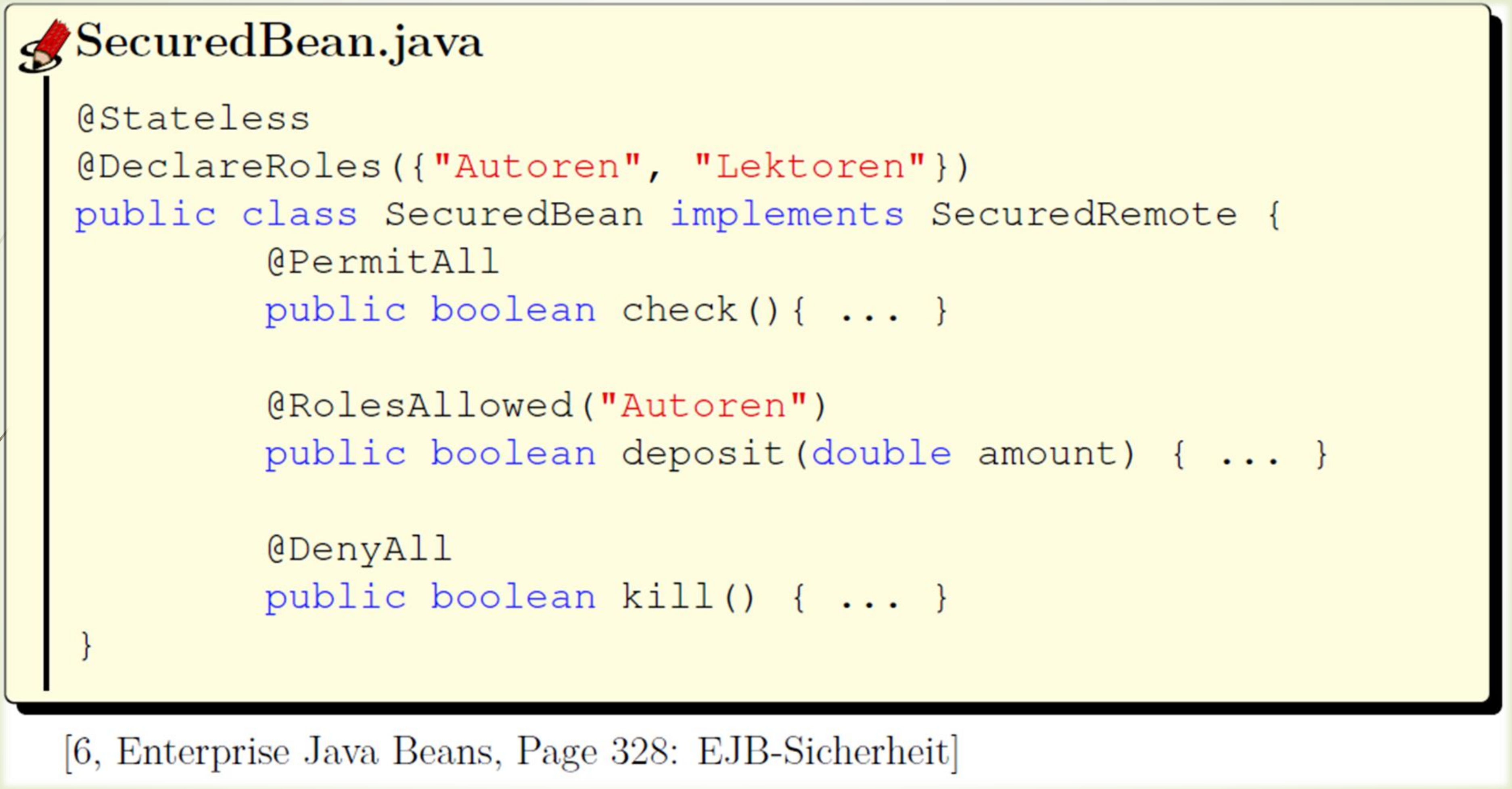
Declarative

- **web.xml**
- Method **permissions**
  - <assembly-descriptor>
- **Altered identity** within security context
  - <security-identity>

Annotations

- **Within Beans**

- @Declare Roles
- @RolesAllowed
- @PermitAll
- @DenyAll



SecuredBean.java

```java
@Stateless
@DeclareRoles({"Autoren", "Lektoren"})
public class SecuredBean implements SecuredRemote {
        @PermitAll
        public boolean check(){ ... }

        @RolesAllowed("Autoren")
        public boolean deposit(double amount) { ... }

        @DenyAll
        public boolean kill() { ... }
}
```

[6, Enterprise Java Beans, Page 328: EJB-Sicherheit]

Programmatic

- **Directly** in Source-Code
  - Within methods of a Bean
- Via **EntityContext**-Object

- **.getCallerPrincipal()**
  - Returns **current principal**
- **Principle**-Object: **.isCallerInRole(<param>)**
  - Returns if the principle is within a certain Role

Application Clients

Security Concerns

- Malware
- Decompiling
- Disassembling
- Custom Clients

Be careful with „trusting“ the client.

Securing Application Clients

- Same authentication requirements as other JEE Components
- Authentication methods are the same as well:
  - HTTP basic authentication
  - HTTP login-form authentication
  - SSL client authentication

Login

- Programmatic
  - EJB Client: ProgrammaticLogin-Class and it‘s methods
  - Server-specific!
- **Frameworks!**
- Login Modules
  - Via Java Authentication and Authorization Service (JAAS)

Digital Certificates

- **Server Authorization**
  - Sometimes the server's identity is important
- **Client Certificates**
  - Self-Authentication
- Secure communication
  - HTTPS
  - SSL

Digital Certificates

Signed by Certificate Authority

- Sometimes identity is **very** important
- E.g. e-commerce

- Create a Certificate
- Let it get signed by a **C**ertificate **A**uthority (**CA**)
  - **VeriSign**
  - **Thawte**

Digital Certificates

Self-Signed

- Sometimes identity is not that important
- But: Secure communication is still required

- In this case:
- Create a Certificate
- Sign it yourself

Digital Certificates **keytool**

- Strong tool that ships with the SDK
- Can be used for certificate creation and signing
- Administration of Public/private keys in general
  - Also: **Client certificates**

Frameworks … - Advantages

- are tested
- are more secure due to public testing
- are supported
- can save time on long term

Frameworks … - Disadvantages

- need to be learned
- can be limited in the possibilities
- need to be trusted

Frameworks

- Provide: Authentication, Authorization, Cryptography
- Apache Shiro
  - Simple Code Structure
  - Not bound to HTTP
- Spring Security
  - Very structured
- JAAS - Java Authentication and Authorization Service
  - Included in Java SE since Java 1.4 (javax.security.auth)

Shiro Example

- **https://github.com/aayvazyan-tgm/JavaEESecurityExample**

Output escaping

- Escape user input
  - To prevent injections
- Escape the output
  - To add an extra layer of security (for the user)
- Do not show Stack traces

- Use a Framework!

Sources

- **JavaOne 2014:** The Anatomy of a Secure Web Application Using Java,
  Shawn McKinney & John Field, September 29, 2014
  San Francisco

- **Java Security:** Sicherheitslücken identifizieren und vermeiden,
  Marc Schönefeld, 1. edition 2011
  Publisher: Hüthig Jehle Rehm GmbH, Heidelberg.
  ISBN/ISSN 978-3-8266-9105-8

- **Enterprise Java Security:** Building Secure J2EE Applications,
  Marco Pistoia, Natara j Nagaratnam, Larry Koved, Anthony Nadalin,
  1. edition 2004
  Publisher: Addison-Wesley Professional.
  ISBN/ISSN: I SBN 0-321-11889-8

- **Official JavaEE Documentation**, Oracle,
  29.09.2014 http://do cs.oracle.com/javaee/7/tutorial/do c/security-intro.htm

- **Java EE 6,**
  Dirk Weil, 1. edition 2012
  Publisher: entwickler.press
  ISBN 978-3-86802-077-9

- **Java EE 6 Cookbook** for Securing, Tuning, and Extending Enterprise Applications,
  Mick Knutson,
  1. edition June 2012
  Publisher: Addison-Wesley Professional.
  ISBN/ISSN: I SBN 9781849683166

Questions?