Java EE7 Security

Gliederung

- Grundlagen
- Wichtige Begriffe
- Sicherheits Mechanismen
- Arten der Sicherheit
- Sicherheit auf Web Ebene
- Sicherheit auf EJB Ebene

Grundlagen

- Authentisierung
- Autorisierung
- Daten Integrität
- Daten Sicherheit
- Nachvollziehbarkeit

Grundlagen

- Authentisierung == Wer bist du?
  - Identifizierung eines Objekts und Überprüfung der Identität
  - Identität entspricht Benutername
  - Identifizierung erfolgt über Credentials
- Autorisierung == Was darfst du?
  - Überprüfung von Berechtigungen eines angemeldeten Benutzers bezogen auf geschützte Ressourcen
  - Berechtigung entsprechen Rollen / Gruppen

Grundlagen

- Daten Integrität

  Stellt die Richtigkeit der Daten fest

- Daten Sicherheit

  Bestimmte Ressourcen sind nur für bestimmte Users zugänglich

- Nachvollziehbarkeit

  Wer hat welche Ressourcen benutzt/verändert

Wichtige Begriffe

- Realms
- User
- Groups
- Roles
- Principal
- Subject
- Session
- Credentials

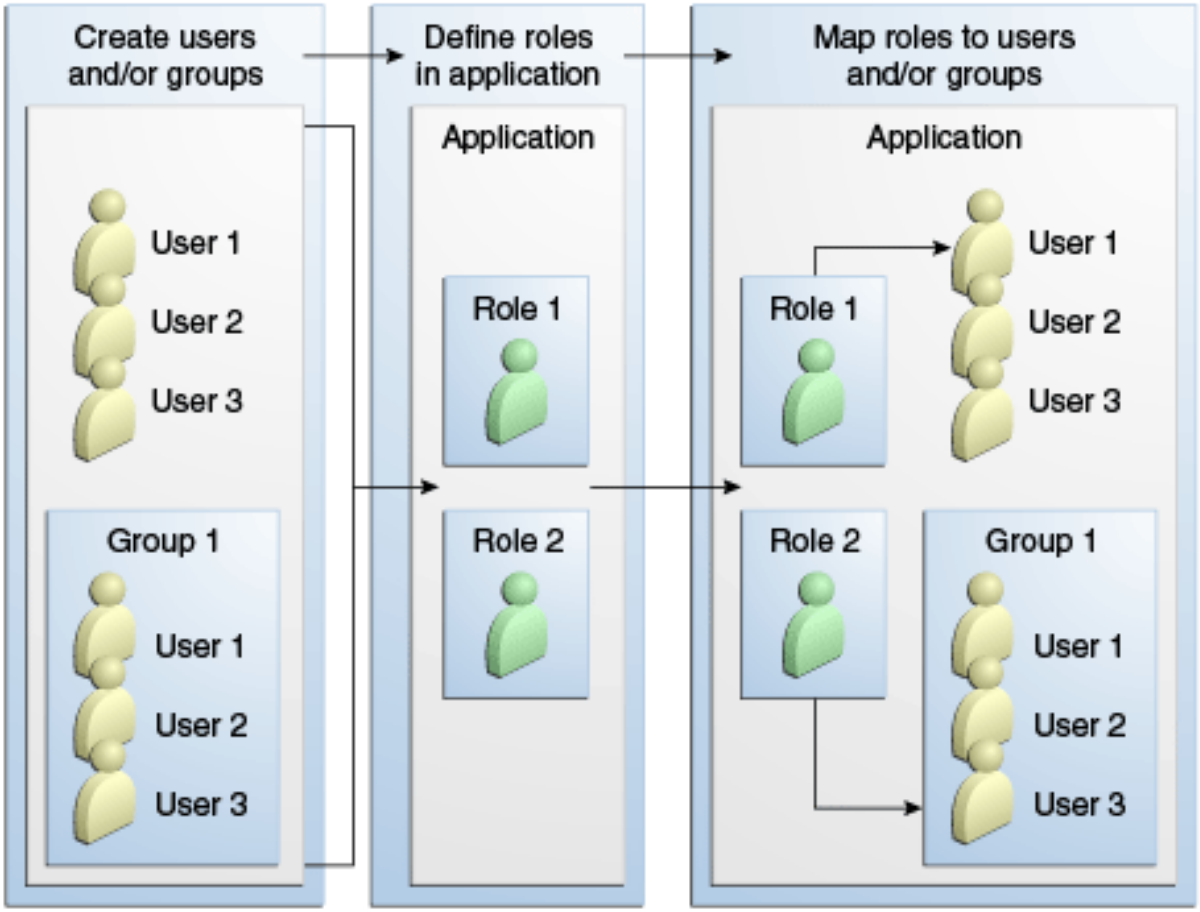
Wichtige Begriffe

- Realms
  - In einem Realm werden die Sicherheitsregeln, für eine Bereich auf einem Server festgelegt
  - Auf einem Server können mehrere Realms definiert werden
- User
  - Ein User ist eine Identität die in einem Realm definiert ist.
  - Ein User entspricht entweder eine Reele Person oder ein Anwendungsprogramm

Wichtige Begriffe

- Groups
  - Eine Group fasst Benutzer mit gleichen Eigenschaften zusammen
- Roles
  - In einer Role ist festgelegt welche Berechtigungen der User oder die Group besitzt

Wichtige Begriffe (Beispiel: User, Groups und Roles)



Wichtige Begriffe

- Principal
  - Ein Principal ist eine Entität, die zur Identifikation genutzt wird.
- Subject
  - Ist eine Sammlung von Informationen zu einer bestimmten Entität

Wichtige Begriffe

- Session
  - In einer Session werden Benutzer Daten in einer „ID“ gespeichert die der Webserver dem Client zuteilt. Bei einer Anfrage des Client, wird dann die „ID“ vom Client zur Autorisierung mitgeschickt
- Credentials
  - Credentials sind Zugriffsberechtigungen

Sicherheits Mechanismen (Java SE)

- Java Authentication and Authorization Servie (JAAS)
- Java Generic Security Service(Java GSS-API)
- Java Cryptography Extension(JCE)
- Java Secure Sockets Extension(JSSE)
- Simple Authentication und Security Layer

Sicherheits Mechanismen (Java EE)

- Application-Layer Security
- Transport-Layer Security
- Message-Layer Security

Sicherheits Mechanismen (Application-Layer Security)

Die Application-Layer Security ist für die Sicherheit in einer Anwendung zuständig und wird auf genau diese Spezifiziert.

Sicherheits Mechanismen (Application-Layer Security)

- Vorteil:
  - Sicherheitsregeln werden genau auf die Sicherheitsbedürfnisse der Anwendung zugeschnitten
- Nachteil:
  - Die Sicherheitsmaßnahmen sind nur für die eine Anwendung brauchbar und kann somit nicht auf andere übertragen werden

Sicherheits Mechanismen (Transport-Layer Security)

Die Transport-Layer Security ist für den Sichern Transport der Daten zwischen Client und Server verantwortlich

- Dies geschieht Mittels Secure Socket Layern(SSL)
- Stellt eine sichere Punkt zu Punkt Verbindung zu verfügung

Sicherheits Mechanismen (Transport-Layer Security)

- Vorteil:
  - Es ist eine relativ leichte Standard Technologie

- Nachteil:
  - Es wird eine Application-Layer Security vorausgesetzt
  - Nur Sicherstellung bis zum Erreichen des Endpunktes

Sicherheits Mechanismen (Message-Layer Security)

Die Message-Layer Security kümmert sich um eine Sichere Ende zu Ende Verbindung

Die Nachricht wird hierbei so verschlüsselt, das Sie nur der gewünschte Empfänger entschlüsseln kann und nicht irgendwelche Zwischenknoten an denen die Nachricht eventuell vorbeigeschickt wird

Sicherheits Mechanismen (Message-Layer Security)

- Vorteil:
  - Sicherstellung das die Nachricht beim gewünschten Empfänger ankommt
  - Kann ohne Probleme über mehrere Zwischenknoten laufen
- Nachteil:
  - Die Einrichtung eines Message-Layer Security ist kompliziert und bringt einiges an Aufwand mit sich

Arten der Sicherheit

- Deklarative Sicherheit

Die deklarative Sicherheit bestimmt die Anforderungen einer Anwendung.

Dies passiert in einer externen XML Datei (Deployment Descriptor) oder über Annotations

- Programmatische Sicherheit

Die programmatische Sicherheit wird genützt, wenn die deklarative Sicherheit nicht mehr ausreicht.

Dies wird Direkt in der Anwendung programmiert.

Sicherheit auf Web Ebene (Deklarativ/Roles)

Festlegung von Sicherheitsrollen durch das security-role und dem role-name Element im Deployment Descriptor

Beispiel: (Auszug aus einem Deployment Descriptor)

```
<security-role>
        <role-name>user</role-name>
</security-role>
```

Sicherheit auf Web Ebene (Deklarativ/Security constraint)

Das Security-constraint Element ist substanziell für die deklarative Sicherheit

Es besteht aus 3 essenziellen Elementen

- web-resource-collection
  - Welche Ressourcen werden beschütz?
- auth-constrain
  - Wer darf auf die Ressourcen zugreifen?
- user-data-constraint
  - Wie sicher soll die Verbindung sein?

Sicherheit auf Web Ebene (Deklarativ/User data constraint)

- In dem user-data-constraint wird mit dem transport-guarantee Element die Sicherheit der Verbindung gewählt
- Es gibt folgende 3 Optionen
  - Confidental (Wandelt automatisch in HTTPS um)
  - Integral
  - None

Sicherheit auf Web Ebene (Deklarativ/Security constraint)

```
<!-- SECURITY CONSTRAINT #1 -->
<security-constraint>
    <web-resource-collection>
        <web-resource-name>wholesale</web-resource-name>
        <url-pattern>/acme/wholesale/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>PARTNER</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Sicherheit auf Web Ebene (Deklarativ/Authentisierung)

Der Authentisierungsmechanismus wird im Deployment Descriptor festgelegt.

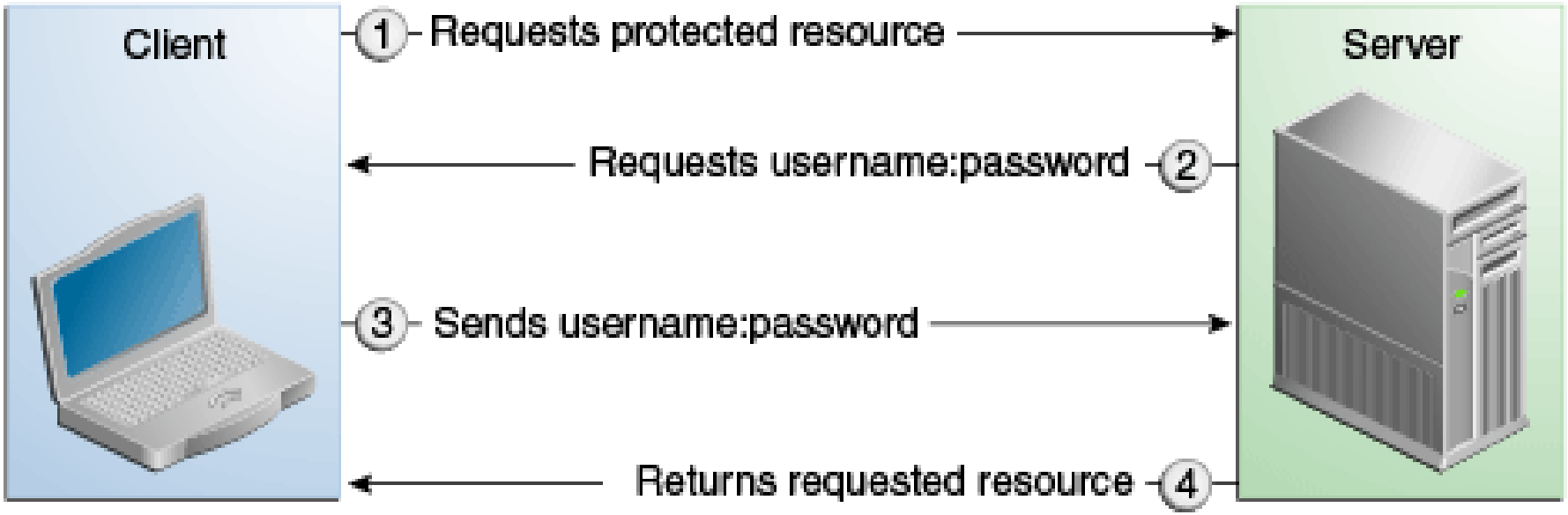
- Basis Authentisierung
- Form basierte Authentisierung
- Digest Authentisierung

Ist kein Authentisierungsmechanismus festgelegt so wird die Basis Authentisierung standardmäßig hergenommen

Sicherheit auf Web Ebene (Deklarativ/Basis Authentisierung)

- Das Basis Authentisierugns Verfahren läuft wie folgt ab:
  - Client sendet eine Anfrage über geschütze Ressourcen
  - Server fordert Anmelde-Daten vom Client
  - Client übergibt Anmelde-Daten
  - Server vergleicht die Daten mit seiner Datenbank und gibt bei erfolgreicher Authentisierung die gewünschten Ressourcen zurück
  - Bei nicht erfolgreicher Authentisierung wird die Standardfehlerseite zurückgegeben (HTTP 404)

Sicherheit auf Web Ebene (Deklarativ/Basis Authentisierung)



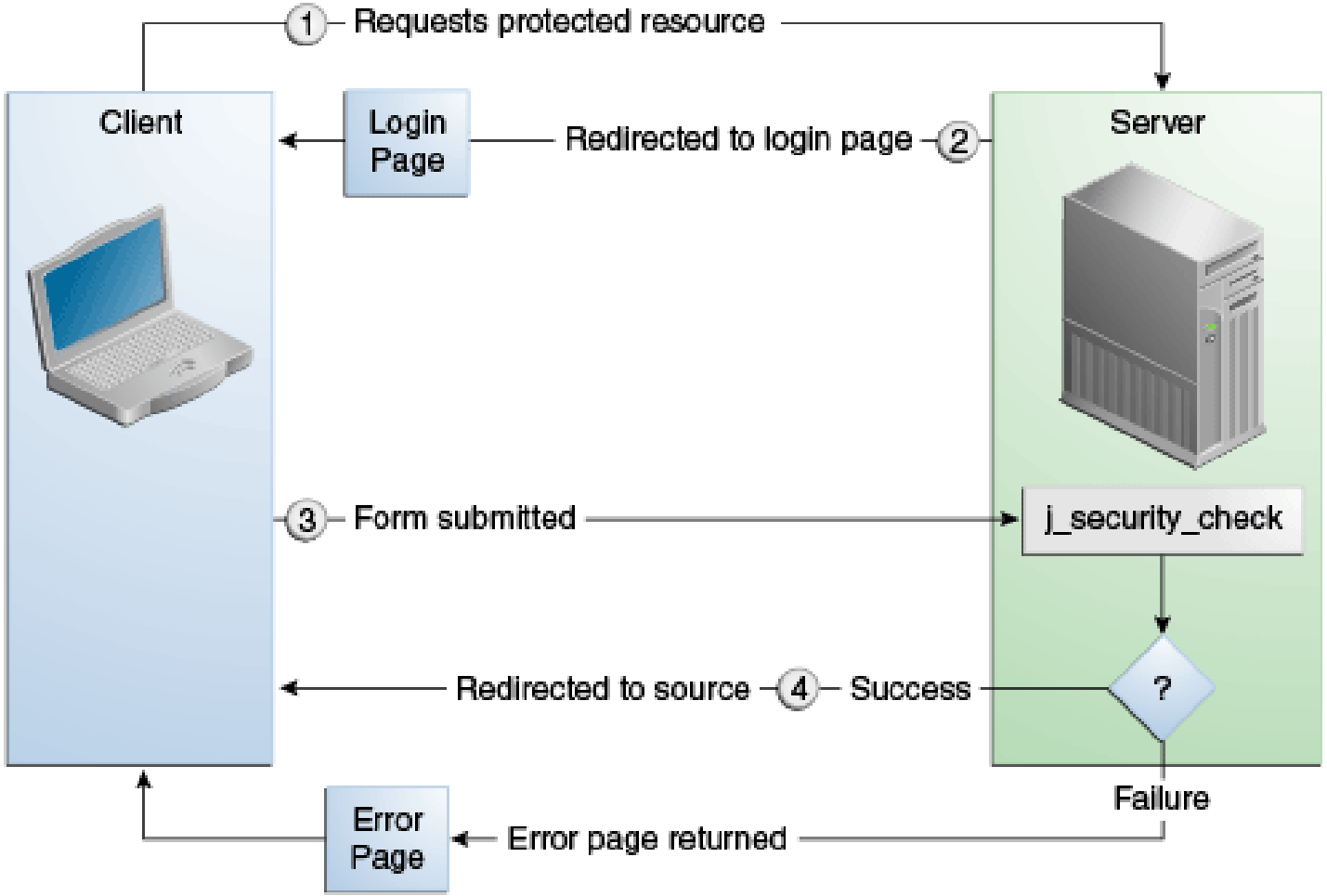
Sicherheit auf Web Ebene (Deklarativ/Digest Authentisierung)

- Das Digest Authentisierungs Verfahren verläuft wie das Basis Verfahren, nur das hier die Anmelde Daten vom Client verschlüsselt werden und von dem Server wieder entschlüsselt werden

Sicherheit auf Web Ebene (Deklarativ/Form Authentisierung)

- Das Form Authentisierungs Verfahren ist ähnlich wie das Basis Verfahren, nur hat kann der Entwickler die Login und die Fehler Seite zu bestimmen und gestalten.

Sicherheit auf Web Ebene (Deklarativ/Form Authentisierung)



Sicherheit auf Web Ebene (Deklarativ/ Login-config Element)

Mit dem login-config Element wird der Authentisierungsmechanismus im Deployment Descriptor gewählt

- Das Login-config Element besteht aus 3 essenziellen Elementen
  - auth-method
  - real-name
  - form-login-config
    - form-login-page
    - form-error-page

Sicherheit auf Web Ebene (Deklarativ/ Login-config Element)

```
<login-config>
 <auth-method>FORM</auth-method>
 <form-login-config>
  <form-login-page>/login.html</form-login-page>
  <form-error-page>/error.html</form-error-page>
 </form-login-config>
</login-config>
```

Sicherheit auf Web Ebene (Programmatisch)

Für die Programmatische Sicherheit auf Web Ebene, wird das HTTP Servlet und die HTTP Servlet Request Klasse benötigt

Sicherheit auf Web Ebene (Programmatisch/HTTP Servlet)

HTTP Servlet als Abstrakte Basisklasse

Überschreibung von den HTTP Methode(doGet, doPost, etc.), die von der Web Anwendung unterstützt werden

(Es muss mindestens eine Methode überschrieben werden)

Diese erwarten ein HTTP Servlet Request Objekt und ein HTTP Servlet Response Objekt

Sicherheit auf Web Ebene (Programmatisch/HTTP Servlet)

```
public class TestServlet extends HttpServlet {
// Bearbeite HTTP GET und POST Methoden
    protected void processRequest(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
```

Sicherheit auf Web Ebene (Programmatisch/HTTP Servlet Request)

- Das HTTP Servlet Request Objekt repräsentiert eine Browser Anfrage
- Das Objekt liefert alle Informationen die vom Browser geschickt werden

Sicherheit auf Web Ebene (Programmatisch/HTTP Servlet Request)

Unter anderem stehen folgende Methoden zur verfügung:

- login / logout
- authenticate
- getRemoteUser
- getUserPrincipal
- isUserInRole

Sicherheit auf Web Ebene (Programmatisch/HTTP Servlet Request)

```
//Code von der Klasse Testservlet
try {
            String userName = request.getParameter("txtUserName");
            String password = request.getParameter("txtPassword");
            try {
                request.login(userName, password);
            } catch(Servlet Expection ex) {
// weitere Code
```

Sicherheit auf EJB Ebene (Deklarativ/ejb-jar.xml)

In der ejb-jar.xml Datei werden die Sicherheitsrollen auf der EJB Ebene festgelegt, die wiederrum in den Klassen durch Annotation zum Einsatz gebracht werden können

- Dies passiert im assembly-descriptor mit dem Element security-role

Sicherheit auf EJB Ebene (Deklarativ/Annotations)

Die Sicherheit auf EJB Ebene kann durch folgende Annotations deklarativ festgelegt werden:

- Declare Roles (Optional, da in der ejb-jar.xml Datei festgelegt)
- RolesAllowed
- PermitAll
- DenyAll

Sicherheit auf EJB Ebene (Programmatisch)

Auf der EJB Ebene kann die Sicherheit auch Programmatisch bestimmt werden

Dafür ist allerdings der Session Context nötig

- Session Context (wird zur Laufzeit abgespeichert)

Benutz die Methode isCallerInRole zur Authentisierung

Benutzt die Methode getCallerPrincipal um den Principal des Callers herauszufinden

Sicherheit auf EJB Ebene (Programmatisch)

```
//benötigte Imports
@Declare Roles(„User“)
Public class testBean
        @Ressource SessionContext ctx;

        @RolesAllowed(„User“)
        public testMethod() {
                Principal callerPrincipal = ctx.getCallerPrincipal();
                If (ctx.getCallerInRole(„User“)) {
                // Rest vom Code
```