Sicherheit in unternehmenskritischen Applikationen

Autor: Manuel Herdt

Wieso ist die Sicherheit von Applikationen so wichtig?

Was verspricht man sich als Unternehmen von den Sicherheitsmaßnahmen?

- Einhalten von Sicherheitsstandards
- Halten von Wettbewerbsvorteilen/-fähigkeit
- Investitionssicherheit
- Risikomanagement
- Vertrauen des Kunden

Sicherheitsziele - VIVA

Grundsätzliche Ziele der Sicherheit:

- Vertraulichkeit
- Integrität
- Verfügbarkeit
- Authentizität

Erstellung eines Sicherheitskonzepts

- Analyse der möglichen Gefahren
- Analyse des Projekts
- Risiko-/Gefahrenanalyse
- Schutzniveau festlegen
- Definition konkreter Ziele
- Umsetzung bei Bedarf
- Dokumentation/Analyses des Ergebnisses

Eigenschaften von Application Security

- **Authentication:** Feststellen der Identität von Server und Client für die Autorisierung
- **Authorization:** Hat der Benutzer die Berechtigung um die Daten oder Operationen auszuführen
- **Data Integrity:** Sind die Daten von einer dritten Partei verändert worden
- **Confidentiality:** Sensitive Daten sollten nur von Berechtigten Benutzern eingesehen werden können

Eigenschaften von Application Security

Non-repudiation: Handlungen von Benutzern können nachvollzogen werden

Quality of Service: Verbesserung der Dienstgüte über verschiedene Technologien

Auditing: Möglichkeiten die Effektivität der Sicherheitsmaßnahmen zu evaluieren

Realms, Users, Groups and Roles

- **Realm:** Ein Realm ist eine für den Server definierte Security Policy Domain und beinhaltet eine Ansammlung von Benutzern die Gruppen zugeteilt sein können.
- **User:** Ein User ist eine Person oder eine Identität einer Applikation. User können Rollen zugewiesen und in Groups eingeteilt werden. Die Authentifikation der User kann über Benutzername/Passwort oder Zertifikat erfolgen.
- **Group:** Eine Group ist eine Liste von authentifizierten Usern die Gemeinsamkeiten aufweisen und aufgrund dieser gleich behandelt werden.
- **Role:** Eine Rolle entspricht einer Zugriffsberechtigung auf gewisse Ressourcen einer Applikation.

Sicherheitsarchitektur Java EE

Die Sicherheit der Applikationsschicht wird von den jeweiligen Container verwaltet!

Unterscheidung:

- Deklarative Sicherheit:
  - Annotationen
  - Deployment Deskriptoren
- Programmatische Sicherheit:
  - Authentisierung/Autorisierung im Code (Geschäftslogik)

Definition der Rollen im JBoss

Definition von Rollen im Properties Files unter „jboss-6.1.0.Final\server\default\conf\props“:

```
<application-policy name="web-console">
    <authentication>
      <login-module
code="org.jboss.security.auth.spi.UsersRolesLoginModule"
      flag="required">
          <module-option name="usersProperties">web-console-
      users.properties</module-option>
        <module-option name="rolesProperties">web-console-
roles.properties</module-option>
      </login-module>
    </authentication>
  </application-policy>
```

Annotationen

Wichtige Annotationen :

Deklaration der Rollen, welche die Klasse verwendet:

- `@DeclareRoles`

Annotationen zur Authorisierung:

- `@RolesAllowed`
- `@PermitAll`
- `@DenyAll`

Es kann immer nur eine der drei verwendet werden!

Annotionen bei Methoden überschreiben Annotationen auf Klassenebene!

Beispiel: Annotations

```
import javax.faces.application.FacesMessage;
...
@ManagedBean(name = "projektManagementBean")
@DeclareRoles ({"Admin","Projektleiter","Geschaeftsfuehrer"})
public class ProjektManagementBean implements
ProjektManagement {
...
@RolesAllowed
({"Admin","Projektleiter","Geschaeftsfuehrer"})
public String getControllingReport () {
...
}
```

Security Constraints

- `<web-resource-collection>` beinhaltet mindestens ein URL Pattern (`<url-pattern>`) auf welches der Zugriff beschränkt werden soll. Optional können Http-Methoden via `<http-method>` bzw. `<http-method-omission>` für das URL pattern restriktiert werden
- `<auth-constraint>` legt fest ob eine Authentifikation notwendig ist um auf die vom URL Pattern beschriebenen Seiten zukommen und welche Rollen zugriffsberechtigung haben
- `<user-data-constraint>` legt fest wie die Datentransfer zwischen Client und Server geschützt ist

Beispiel: Security Constraints

```
<!-- SECURITY CONSTRAINT  -->
<security-constraint>
  <display-name>Projektmanagement</display-name>
  <web-resource-collection>
    <web-resource-name>projektmanagement</web-resource-name>
    <url-pattern>/projekt/management/*</url-pattern>
    <http-method-omission>GET</http-method-omission>
    <http-method-omission>POST</http-method-omission>
  </web-resource-collection>
  <auth-constraint>
    <role-name>admin</role-name>
    <role-name>projektleiter</role-name>
     <role-name>geschaeftsfuehrer</role-name>
    </auth-constraint>
    <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

Programmatische Sicherheit

Rollenabfragen in der Geschäftslogik:

HTTPServletRequest:

- getUserPrincipal()
- isUserInRole(String role)

EJBContext:

- getCallerPrincipal()
- isCallerInRole(String role)

Vorausetzungen: Rollen via Annotationen und Deployment Deskriptor definiert

Beispiel:

```
@RolesAllowed
({"Admin","Projektleiter","Geschaeftsfuehrer"})
public String getControllingReport () {
    // Erstellen des Reports
    ...
    if(context.isCallerInRole(„Geschaeftsfuehrer“)) {
        //  sensible Daten hinzufügen
        ...
    }
...
}
```

Authentifizierungs-mechanismen

Java EE unterstützt fünf Authentifizierungs-mechanismen:

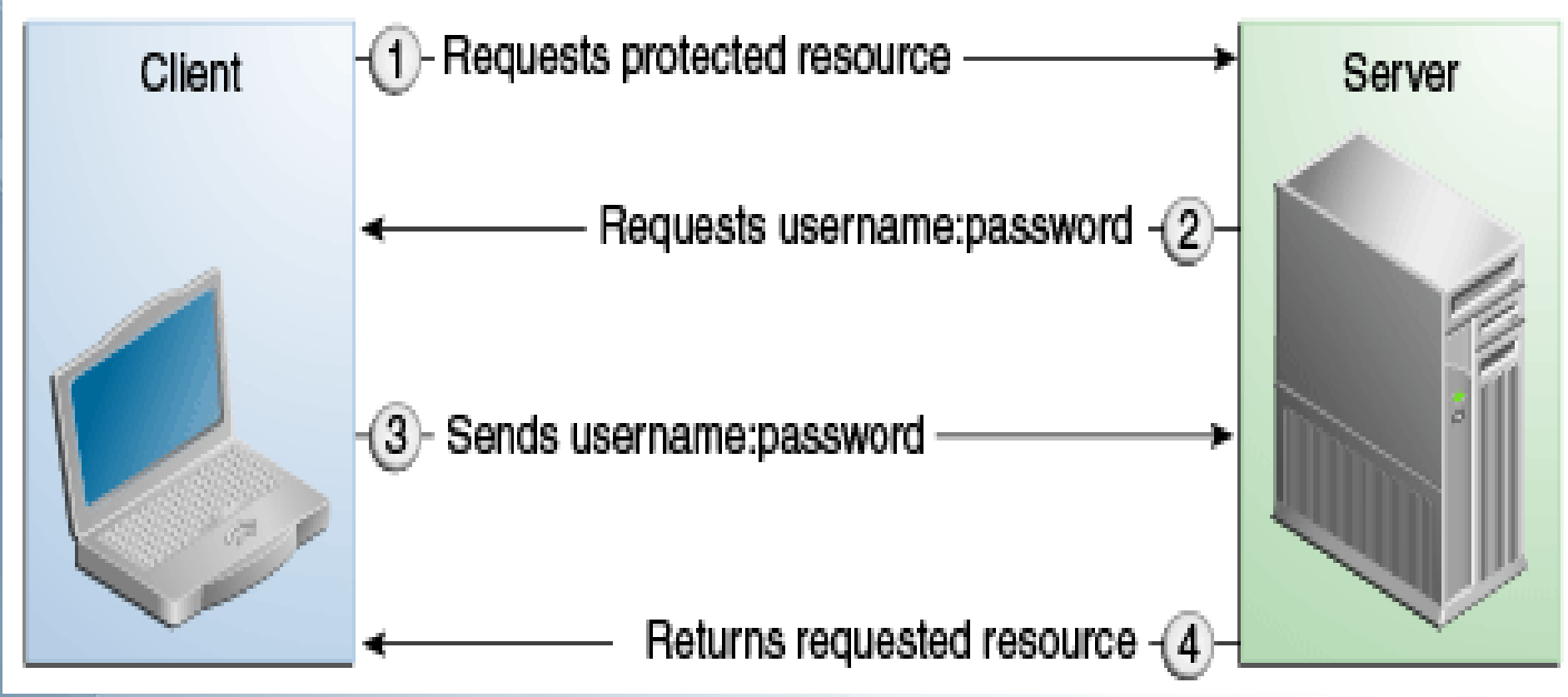
- Basic Authentication
- Form-Based Authentication
- Digest Authentication
- Client Authentication
- Mutual Authentication

HTTP Basic Authentication

Der Ablauf einer Basic Authentication sieht wie folgt aus:

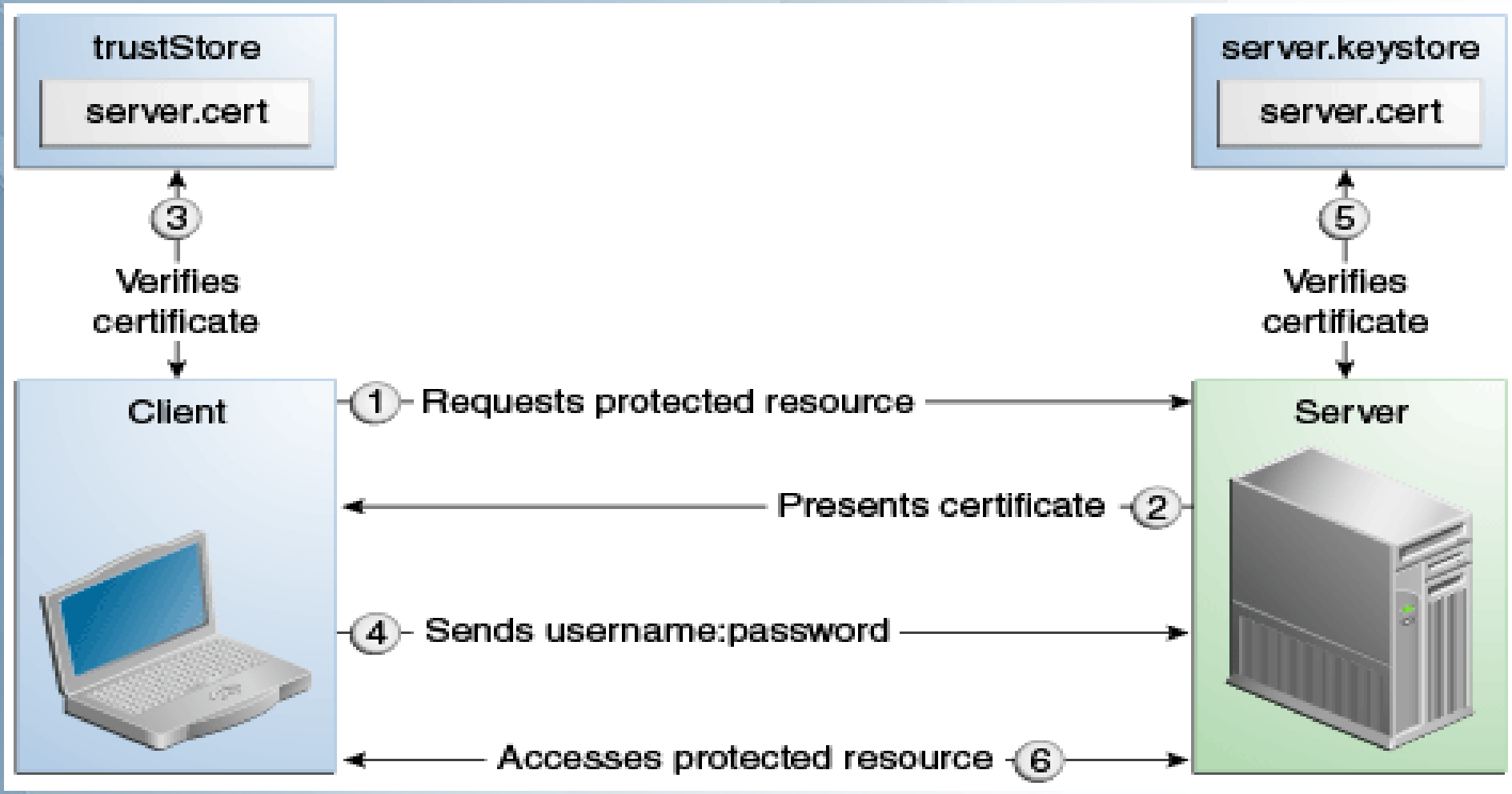
1. Der Client sendet ein Anfrage
2. Der Web Server verlangt nach Benutzername und Passwort
3. Der Client sendet den Benutzernamen sowie das dazugehörige Passwort an den Web Server
4. Ist die Authentifizierung erfolgreich sendet der Web Server die angeforderten Ressourcen an den Client

HTTP Basic Authentication



Quelle: Java EE 7 Tutorial 48.2.2.1 HTTP Basic Authentication

Mutual Authentication



Quelle: Java EE 7 Tutorial 50.2.2 Mutual Authentication

Vielen Dank für ihre Aufmerksamkeit.

Fragen?