



Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences



CAS Enterprise Application Development Java EE

Modul Java EE Security

► Peter Andres - ISC-EJPD

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement EJPD
Informatik Service Center ISC-EJPD
Abteilung Technologie
Peter Andres

Zu meiner Person



Name, Vorname:
Kontaktangaben:

Peter Andres
ISC-EJPD
Güterstrasse 24, 3003 Bern
✉ peter.andres@isc-ejpd.admin.ch
☎ +41 (0)58 46 37872

Funktion beim ISC-EJPD Abteilungsleiter Technologie (CTO)

Letzte Ausbildungen

CAS Information Security (fhnw)
Certified Information Systems Security
Professional (CISSP)
CAS Security & Privacy (bfh)
MAS-IT (bfh)

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
2



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement EJPD
Informatik Service Center ISC-EJPD
Abteilung Technologie
Peter Andres

Erwartungen an den Kurs



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
3



Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

Lernziel

Ziel dieses Kurses ist es, die grundlegenden Sicherheitstechniken, welche die JEE 7 Plattform bietet, zu verstehen und in der Praxis anwenden zu können

Lerninhalte

- 1 Einleitung
Überblick
- 2 Grundlagen
Einfache Beispiel
- 3 Grundlagen
Begriffe
- 4 Sicherheit im
Web-Tier
- 5 Sicherheit im
Web-Tier (Advanced)
- 6 Sicherheit im
EJB-Tier
- 7 Sicherheit
Webservices
- 8 Ausblick JEE8
(Sicherheit)
- 9 Transportsicherheit
Secure Sockets Layer
- 10 Übungen
Bookstore

1

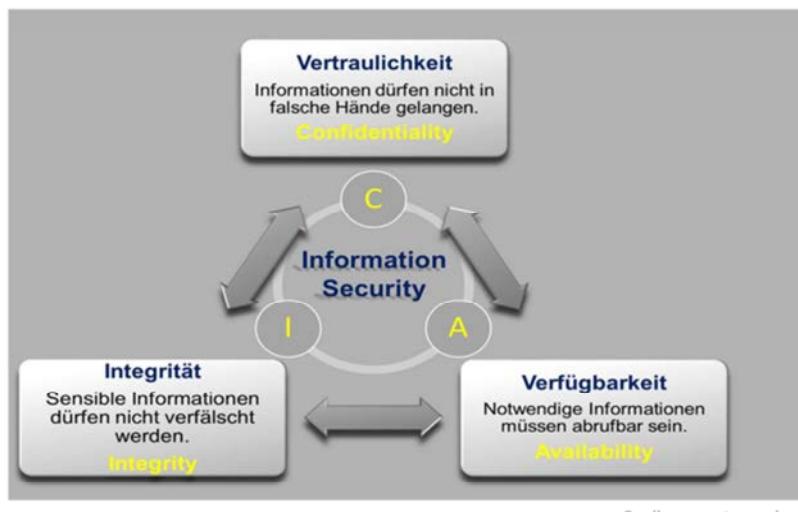
Einleitung Überblick und Einführung

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
6



Eidgenössische Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

Java EE 7 Security Einleitung: C.I.A. Prinzip / Dreieck



Quelle comupterwoche

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

7

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Das **Confidentiality, Integrity, Availability** Prinzip (deutsch: Vertraulichkeit, Integrität, Verfügbarkeit) sind in der IT-Sicherheit die drei wesentlichen Grundbedrohungen. Zu diesen drei Bedrohungen zählen der Verlust der Verfügbarkeit, der Integrität und der Vertraulichkeit von Daten.
- **Vertraulichkeit** (englisch: confidentiality): Vertraulichkeit ist der Schutz vor unbefugter Preisgabe von Informationen (innerhalb von Systemen und bei der Übertragung von Daten).
- **Integrität** (englisch: integrity): Integrität bezeichnet die Sicherstellung der Korrektheit (Unversehrtheit) von Daten (innerhalb von Systemen und bei der Übertragung von Daten).
- **Verfügbarkeit** (englisch: availability): Die Verfügbarkeit von Services, Funktionen eines IT-Systems, einer IT-Anwendungen oder IT-Netzen oder auch von Informationen ist zur gewünschten Zeit vorhanden.

Erweiterte Schutz-Ziele

- **Authentizität** (englisch: authenticity) Mit dem Begriff Authentizität wird die Eigenschaft bezeichnet, die gewährleistet, dass ein Kommunikationspartner tatsächlich derjenige ist, der er vorgibt zu sein.
- **Nichtabstreitbarkeit** (englisch: non repudiation): Hierbei liegt der Schwerpunkt auf

der Nachweisbarkeit gegenüber Dritten. Ziel ist es zu gewährleisten, dass der Versand und Empfang von Daten und Informationen nicht in Abrede gestellt werden kann.

(siehe auch BSI

https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/Glossar/glossar_node.html)

Java EE 7 Security Einleitung: Integrale Sicherheit



Quelle IOS VBS



Quelle linux-ag

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
8

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

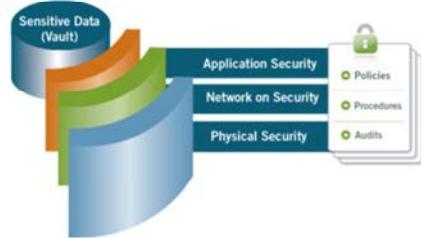
Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Nur die integrale Sicherheit bringt **wirkliche und integrale** Sicherheit (d.h. alle Aspekte müssen berücksichtigt sein)
- Java EE7 Security deckt nur einen kleinen (aber wichtigen) Teil der Informations- resp. IT-Sicherheit ab
- Einen einfachen Überblick liefert das VBS in einem Video
<https://www.youtube.com/watch?v=q6J7fTZ9R90>

Java EE 7 Security Einleitung: Defense in depth



Quelle nsslabs



Quelle nsslabs

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
9



Eidgenössisches Justiz- und Polizeidepartement EJPO
Informatik Service Center ISC-EJPO
Abteilung Technologie
Peter Aebi

Java EE 7 Security Einleitung: Kleines Beispiel

Wie sicher ist eine ...



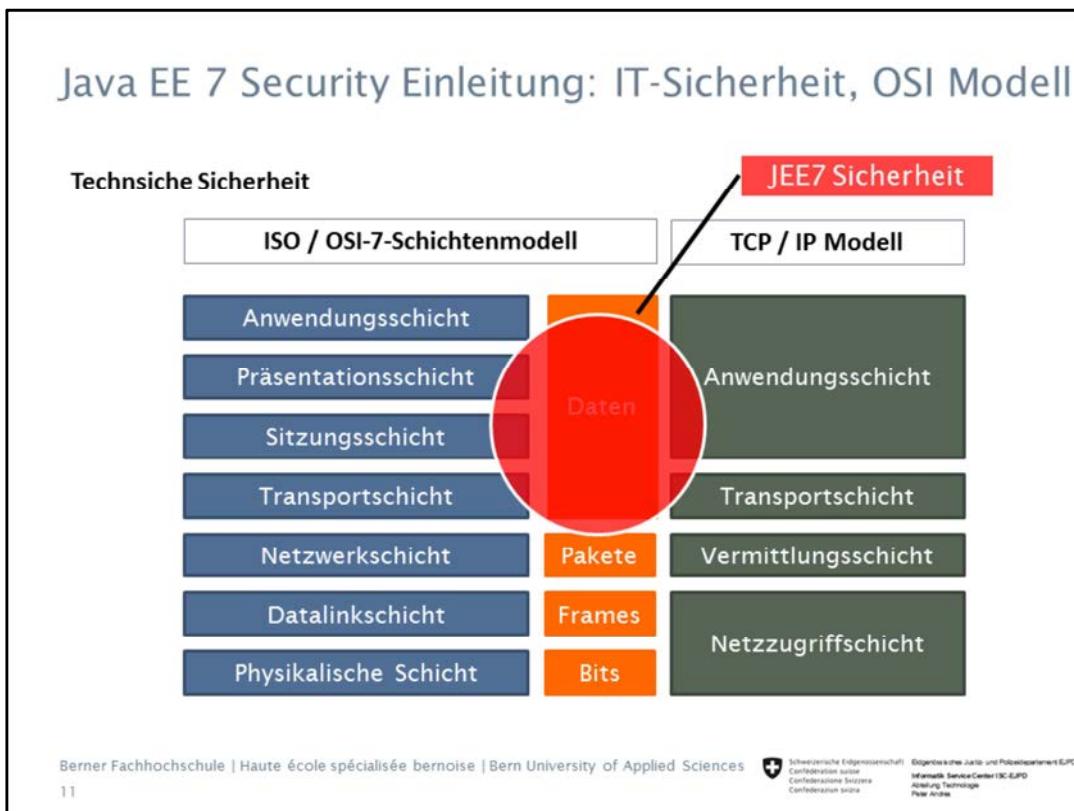
Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
10



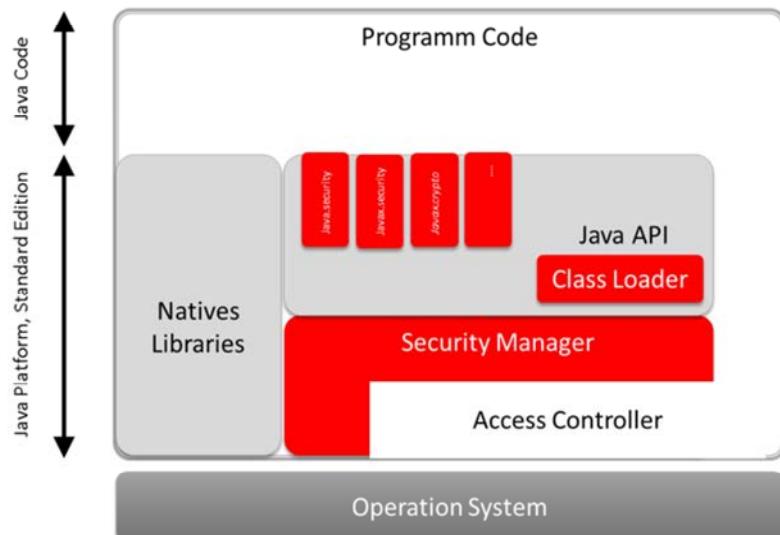
Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

- Siehe auch Artikel des CT's 2014

Java EE 7 Security Einleitung: IT-Sicherheit, OSI Modell



Java EE 7 Security Einleitung: Java SE Security



- Bereits die Java Standard-Edition (JSE) ist ausgerüstet mit vielen Sicherheitsfeatures und Sicherheits-API wie z.B. der Security-Manager oder der Classloader.
- Dieser JEE 7 Sicherheit Kurs behandelt **keine** JSE Sicherheits-APIs – JEE 7 Security basiert jedoch (teilweise) auf diesen.

Java EE 7 Security Einleitung: Java 7 SE Security: Links

- ▶ **Allgemeines über Java SE Security**
 - <http://docs.oracle.com/javase/7/docs/technotes/guides/security/>
 - <http://docs.oracle.com/javase/tutorial/essential/environment/security.html>
- ▶ **Java Security-Manager (gute Übersicht, Fachhochschule Nordwestschweiz)**
 - <http://gruntz.ch/courses/sem/ss03/JavaSecurityManager.pdf>
- ▶ **Java Classloader (gute Übersicht, Fachhochschule Nordwestschweiz)**
 - <http://www.gruntz.ch/courses/sem/ws03/ClassLoaders.pdf>

- JCE (Java Cryptography Extension) ist ein häufig verwendetes API – z.B. für die Berechnung eines digitalen Fingerabdruckes eines Passwortes (Message-Digest) wie folgendes Beispiel zeigt:

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
byte[] hash = digest.digest(text.getBytes("UTF-8"))
```

- JSSE wird ebenfalls häufig verwendet – z.B. um SSL Verbindungen aufzubauen wie folgendes Beispiel zeigt.:

```
Security.addProvider( new com.sun.net.ssl.internal.ssl.Provider());
SSLocketFactory factory = (SSLocketFactory)SSLocketFactory.getDefault();
SSLocket socket = (SSLocket)factory.createSocket(url.getHost(), 443);
```

Java EE 7 Security Einleitung: Java 7 SE Security: Links

► JCE (Java Cryptography Extension)

- JCE ist ein Java-API und Framework für kryptographische Aufgaben wie Verschlüsselung, Message-Digest, Authentisierung und Schlüsselverwaltung.
- <http://docs.oracle.com/javase/7/docs/technotes/guides/security/>

► JSSE (Java Secure Socket Extension)

- Java JSSE ist ein Java-API für Secure Sockets Layer (SSL).
- <https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html>

► SASL (Simple Authentication and Security Layer)

- SASL ist ein Framework, das von verschiedenen Protokollen zur Authentisierung verwendet werden kann
- <https://docs.oracle.com/javase/7/docs/technotes/guides/security/sasl/sasl-refguide.html>

- JCE (Java Cryptography Extension) ist ein häufig verwendetes API – z.B. für die Berechnung eines digitalen Fingerabdruckes eines Passwortes (Message-Digest) wie folgendes Beispiel zeigt (Hash erzeugen eines Passwortes):

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
byte[] hash = digest.digest(text.getBytes("UTF-8"))
```

- JSSE wird ebenfalls häufig verwendet – z.B. um SSL Verbindungen aufzubauen wie folgendes Beispiel zeigt.:

```
Security.addProvider( new com.sun.net.ssl.internal.ssl.Provider());
SSLContextFactory factory =
(SSLContextFactory)SSLContextFactory.getDefault();
SSLContext socket = (SSLContext)factory.createSocket(url.getHost(), 443);
```

Java EE 7 Security Einleitung: OWASP Top 10

JEE Security

Applikation-Security

OWASP Top 10 – 2010 (alt)	Δ	OWASP Top 10 – 2013 (neu)
A1 – Injection	=	A1 – Injection
A3 – Fehler in Authentifizierung und Session-Management	↗	A2 – Fehler in Authentifizierung und Session-Management
A2 – Cross-Site Scripting (XSS)	↘	A3 – Cross-Site Scripting (XSS)
A4 – Unsichere direkte Objektreferenzen	=	A4 – Unsichere direkte Objektreferenzen
A6 – Sicherheitsrelevante Fehlkonfiguration	↗	A5 – Sicherheitsrelevante Fehlkonfiguration
A7 – Kryptografisch unsichere Speicherung – mit A9 →	↗	A6 – Verlust der Vertraulichkeit sensibler Daten
A8 – Mangelhafter URL-Zugriffsschutz – erweitert zu →	↗	A7 – Fehlerhafte Autorisierung auf Anwendungsebene
A5 – Cross-Site Request Forgery (CSRF)	↘	A8 – Cross-Site Request Forgery (CSRF)
<Teil von A6: Sicherheitsrelevante Fehlkonfiguration>	neu	A9 – Verwendung von Komponenten mit bekannten Schwachstellen
A10 – Ungeprüfte Um- und Weiterleitungen	=	A10 – Ungeprüfte Um- und Weiterleitungen
A9 – Unzureichende Absicherung der Transportschicht	↗	Zusammen mit 2010-A7 nun im neuen 2013-A6

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
15



Eidgenössische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement (EJPO)
Informatik Service Center (ISC-EJPO)
Abteilung Technologie
Peter Andri

- Das Open Web Application Security Project hat eine neue Ausgabe der OWASP Top 10 (2013) vorgelegt. Die erstmals vor zehn Jahren veröffentlichte Rangliste der zehn schwerwiegendsten Sicherheitsschwachstellen von Webanwendungen genießt unter Sicherheitsexperten und Webentwicklern einen hohen Stellenwert. Für die neue Rangliste wurden über 500.000 Schwachstellen in mehreren hundert Unternehmen und Tausenden Applikationen beobachtet.
- OWASP Top 10: https://www.owasp.org/index.php/Top_10_2013-Top_10 ()

Java EE 7 Security Einleitung: Heartbleet Lücke ...

Einfach erklärt

<http://www.heise.de/security/artikel/So-funktioniert-der-Heartbleed-Exploit-2168010.html>

Demo

<https://www.youtube.com/watch?v=jaobW2hd6Ho>

<https://www.youtube.com/watch?v=OMtvF-FTxGQ>

<https://www.youtube.com/watch?v=hTK0pywfmDE>

2

Grundlagen Einfaches Beispiel

Java EE 7 Security: Ein einfaches Beispiel, Schritt 1

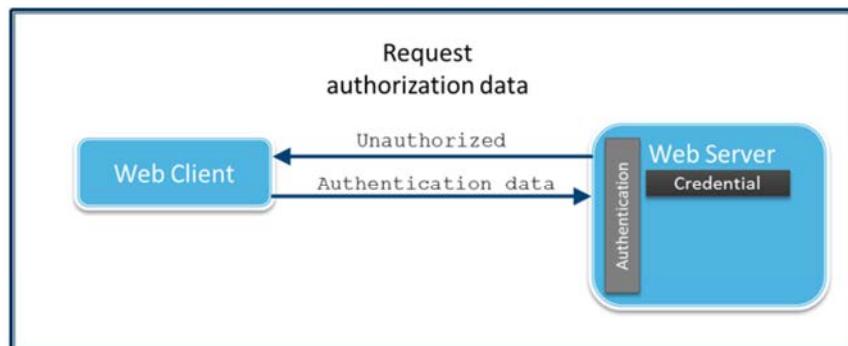
Initialer (erstmaliger) Request



- Kleines Beispiel zu Beginn: Ein Webclient will auf eine geschützte (Web-) Ressource zugreifen und anschliessend ein «Businesscall» auf eine geschützte Businesskomponente machen.
- Der Webclient ist (noch) nicht authentisiert
- Der Webclient macht einen (initialen) Request auf die geschützte Ressource.
- Der Webserver detektiert den nicht authentisierten Client und aktiviert den (konfigurierten) Authentisierungs-Mechanismus

Java EE 7 Security: Ein einfaches Beispiel, Schritt 2

Initiale Authentisierung des Web Client (Client)



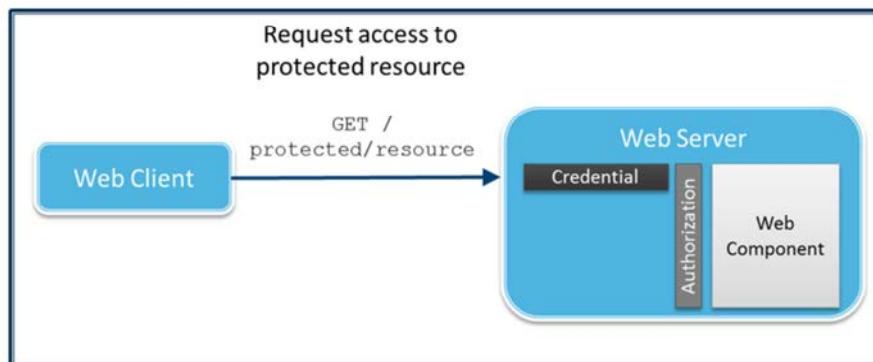
Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
19

 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Der Webserver signalisiert dem Webclient, dass er sich zuerst authentisieren muss, bevor er auf die geschützte Ressource zugreifen darf.
- Der Client sendet die Authentisierungsdaten zum Webserver – das können z.B. Username / Password sein.
- Beliebte Authentisierungsverfahren sind
 - Username / Passwort
 - (Client) Zertifikate (public key / private key) – z.B SuisseID
 - Onetime – Password (RSA – Token, Streichliste, usw.)
 - Challenge – Response
 - Biometrische Verfahren
- Der Webserver validiert die Authentisierungsdaten und / oder delegiert diese weiter an einen entsprechenden Sicherheitsdienst.
- Wenn die Validierung erfolgreich ist, setzt der Webserver einen Identitäsnachweis (das Credential) für den Benutzer (Webclient).
- Die letzten beiden Punkte beschreiben die Authentisierung des Webclient

Java EE 7 Security: Ein einfaches Beispiel, Schritt 3

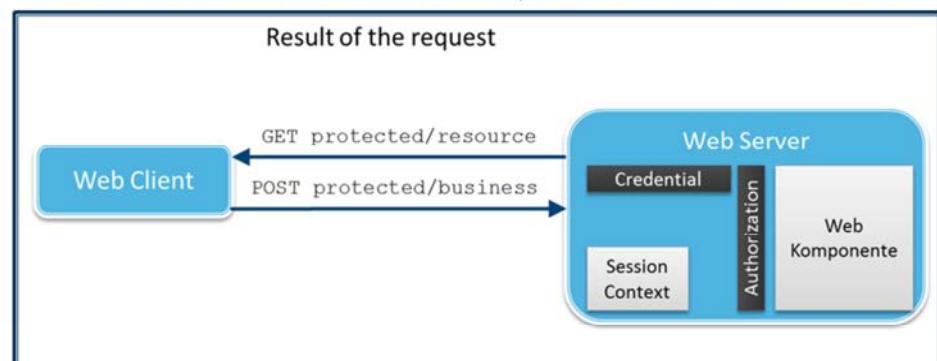
Autorisierung des Web Client



- Der Webserver verwendet nun den Identitäts-Nachweis (**das Credential**) des Benutzers bei weiteren Zugriffen auf geschützte Ressourcen.
- Der Webserver verwendet nun die der geschützten Ressource zugeordneten Sicherheitseinstellungen (security policy), um festzustellen, welche Rollen (User mit Rollen) Zugriff haben auf die geschützte Ressource (diese Sicherheitseinstellungen können per Annotationen oder im web.xml erfolgen).
- Der Webserver prüft nun ob das (Benutzer-) Credential auf eine Rolle «gemapped» werden kann, welche in den Sicherheitseinstellungen konfiguriert wurde
- Ist die Rolle berechtigt auf die Ressource zuzugreifen, liefert der Webserver diese Ressource aus (Autorisierung)
- Die letzten beiden Punkte beschreiben die Autorisierung des Webclient

Java EE 7 Security: Ein einfaches Beispiel, Schritt 4

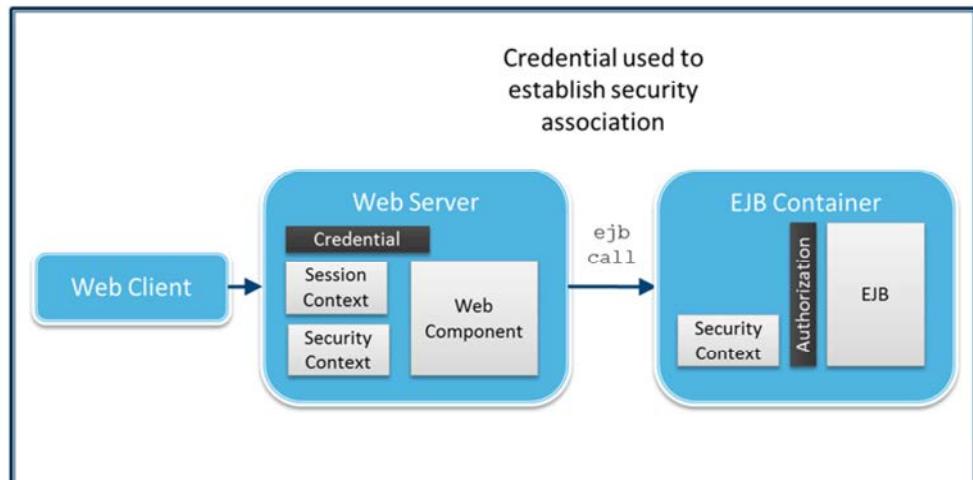
Geschützte Ressource ausliefern, Business-Call ausführen



- Wenn der Benutzer authentisiert und autorisiert wurde, liefert der Webserver die angeforderte, geschützte Ressource aus.
- Im Beispiel kann nun der Webclient Business-Daten (Business-Call) an die Applikation senden (z.B. per Formular)

Java EE 7 Security: Ein einfaches Beispiel, Schritt 5

Businesscall ausführen



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
22

 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössische Technische Hochschule
Informatik-Service-Center (ISC-EIPO)
Abteilung Technik
Peter Andri

- Wenn der Webclient nun einen Business-Call ausführt, dann verwendet der Webserver die Credentials des Benutzers.
- Die Benutzer Credentials werden vom Webserver dem EJB Container propagiert (innerhalb des GIOP [General Inter-ORB Protocol] Protokolls).
- Genauer: CSIV2 (Common Secure Interoperability Version 2) innerhalb von GIP ermöglicht es Authentisierungs- und Autorisationsdaten zu transportieren.
- Der Webserver und der EJB Container haben einen "Security Context", welcher in einer Relation zueinander stehen.
- Der EJB Container ist verantwortlich, zu kontrollieren, ob der User (in seiner Rolle) autorisiert ist, die Business-Methode auszuführen oder nicht.
- Wie der Webcontainer, nimmt der EJB Container das Credential resp. dessen Rolle und überprüft, ob die entsprechende Methode für die Rolle autorisiert ist oder nicht.
- Die Sicherheitseinstellungen im EJB Container können per Annotationen oder im Deploymentdeskriptor (ejb-jar.xml) erfolgen.
- Befinden sich Webserver und EJB Container im selben Applikationsserver, erfolgt im EJB Container keine Authentisierung mehr (Glassfish).

Java EE 7 Security: Ein einfaches Beispiel, Demo

In der Praxis sieht das dann folgendermassen aus ...



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
23



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

- Das einfache Beispiel kann unter «JEE7_Security_Kurs_Sample.zip» in NetBeans importiert und getestet werden
- Dazu muss im Glassfish Server unter «server-config/Sicherheit/Realms» ein neuer Realm namens «SecureRealm» konfiguriert werden (! JAAS-Kontext = fileRealm)
- Anschliessend muss ein User in der Gruppe USER angelegt werden (siehe auch Übung 1)
- Analyse des Web-Traffic mit Firefox, dabei die Entwicklungstools (Netzwerkanalyse) einschalten
- Decodieren des Base64-Headers kann unter «<https://www.base64decode.org/>» gemacht werden. Vorsicht – niemals produktive Daten und oder Passwörter online decodieren.

Java EE 7 Security : Das Ziel



- Ziel des Kurses ist es, diese in dem vorherigen Beispiel gezeigten Mechanismen und Zusammenhänge im Detail zu verstehen und anwenden zu können.

3

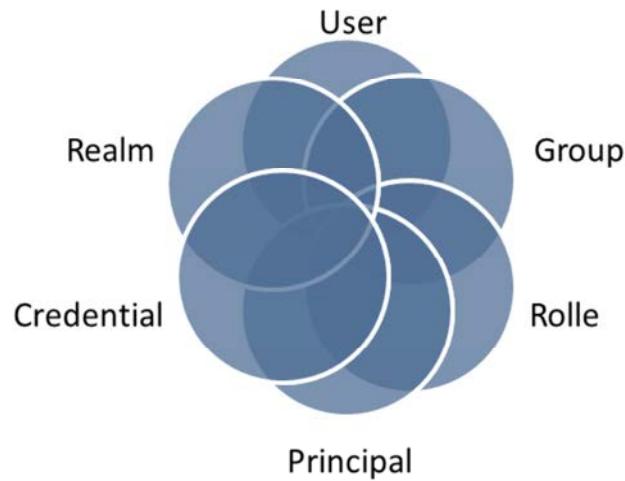
Grundlagen Begriffserklärung

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
25



Eigentum des Justiz- und Polizeidepartement (JPO)
Internat. Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

Java EE 7 Security – Grundlagen: Begriffserklärung



Java EE 7 Security – Grundlagen: Begriffserklärung

User

Ein Benutzer ist eine Identität, welche im Identity-Store (z.B. LDP, Datei usw.) angelegt und definiert wurde. Ein Benutzer kann eine oder mehrere Rollen einnehmen und kann einer Gruppe zugeordnet werden.

Group

Eine Gruppe besteht aus einem Set von Usern, mit gemeinsamen Merkmalen, die in der Regel über eine Reihe von gemeinsamen Berechtigungen verfügen. Eine Gruppe kann eine oder mehrere Rollen einnehmen.

Rolle

Eine Rolle ist ein abstrakter Name für eine Berechtigung einer geschützten Ressource innerhalb einer Applikation. Jede Rolle kann einem oder mehreren Benutzern und / oder Gruppen zugewiesen werden.

- A **User** is an individual identity which is defined in the identity storage. The storage which can be an RDBMS, flat file or LDAP server contains user attributes like username and password.
- A **Group** is a set of users classified with a set of common characteristics which usually lead to a set of common permissions and access levels.
- A **Role** is a Java EE concept to define access levels. A Java EE application developer specifies which roles can access which set of the application functionalities. These roles are then mapped to users and groups using the vendor specific configuration files.

Java EE 7 Security – Grundlagen: Begriffserklärung

Credential

Ein Credential ist ein Identitätsnachweis, der einem System die Identität eines Benutzers (oder eines Systems) bestätigt. Dies können Ausweispapiere, Passwörter, Ergebnisse von kryptografischen Verfahren usw. sein.

Principal

Ein Principal ist eine Identität, die anhand eines bekannten Credential authentisiert werden kann.

- A **Principal** is an identity with known credentials which can be authenticated using an authentication protocol.
- A **Credential** contains or references information used to authenticate a principal for Java EE product services. Password is a simple credential used for authentication.
- Different application servers use different methods to map users, groups and roles to each other.

Java EE 7 Security – Grundlagen: Begriffserklärung

Security Realm

Ein (Security) Realm ist eine definierte Sicherheits-Domäne innerhalb eines Web- oder Applikationsservers. Zu dieser Domäne können verschiedene Benutzer und Gruppen von Benutzern aus unterschiedlichen Identity-Stores (z.B. LDAP) hinterlegt werden.

Jede abgesicherte Applikation wird genau mit einem solchem Realm verlinkt und holt sich über diesen die Benutzerdaten zur Validierung.

Die Web- und Applikationsserver bieten meist verschiedene Möglichkeiten zur Definition (Dateien, Datenbanken, LDAP usw.). So ist es z.B. fast immer möglich, Benutzer und Gruppen mittels Key-Value-Dateien zu definieren.

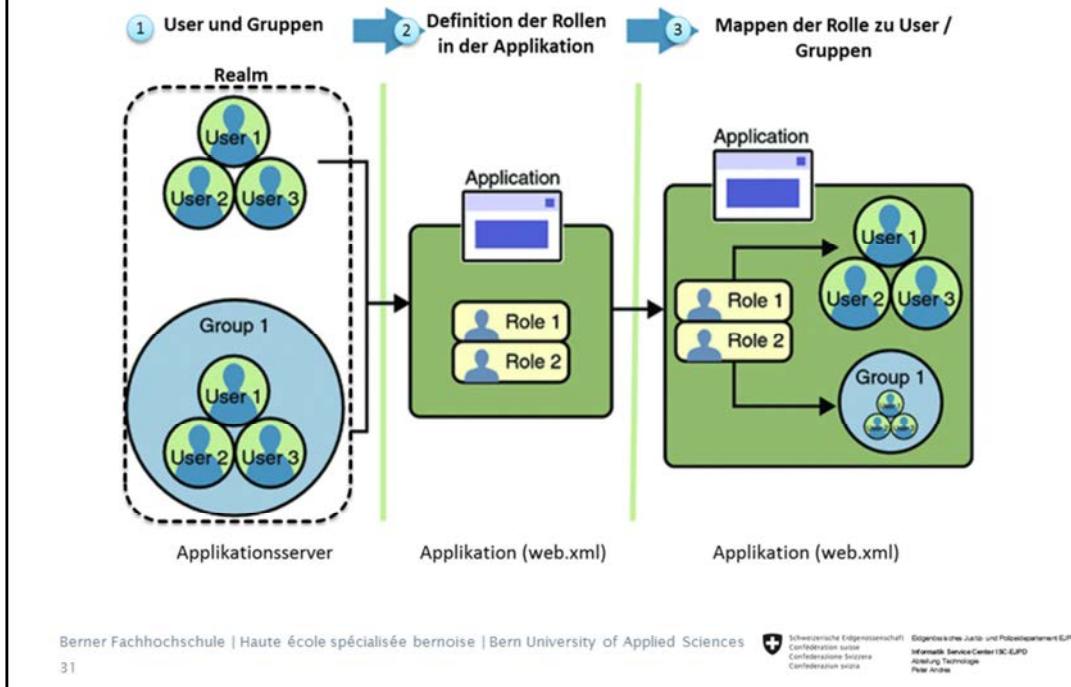
Java EE 7 Security – Grundlagen: Basis – «Rollen»

Rollen-Definition gemäss Oracle (im JEE7 Kontext)

- ▶ Der Applikationsentwickler definiert, wie User authentisiert werden sollen (Definition der LOA – Level of Assurance, z.B. mit Username / Passwort)
- ▶ Der Applikationsentwickler definiert welche Ressourcen der Applikation zu schützen sind (mit Hilfe von Annotationen, Deskriptoren oder programmatisch)
- ▶ Der Administrator erstellt im Applikationsserver (oder in einem Identity-Store) Benutzer und Gruppen
- ▶ Der Applikationsdeployer ordnet Rollen den Usern und / oder Gruppen zu. Der Applikationsserver mapped Rollen zu den Usern und Gruppen

- Die Definition der JEE Rollen von Oracle wird in der Praxis selten so gelebt

Java EE 7 Security – Grundlagen: Benutzer Mapping



In der grafischen Übersicht:

1. Definition von User und Gruppen im Applikationsserver [Rolle Administrator]
2. Definition der Rollen und des Zugriffs-Schutz innerhalb der Applikationen [Rolle Applikationsentwickler]
3. Zuordnen (mappen) der Rollen zu den Usern und / oder Gruppen [Rolle Applikationsentwickler / Applikationsdeplayer]

Java EE 7 Security – Grundlagen: Praxisbezug

Hier am Beispiel des Glassfish-Application-Server 4.1

The screenshot shows the Glassfish Administration Console interface. It is divided into several panels:

- «File-based» Realm**: Shows the configuration of a 'Securrealm' realm, including 'Realms', 'JAAS-Konten', 'Gruppen', and 'Benutzer' (Users).
- User und Gruppen**: Shows the 'User' configuration for the 'Securrealm'.
- Group / Role mapping**: Shows the 'Group / Role mapping' configuration for the 'Securrealm'.
- glassfish-web.xml**: Shows the XML configuration for the 'Securrealm' in the 'glassfish-web.xml' file.

The 'glassfish-web.xml' content is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD
glassfish-web-app error-url//">
<context-root>/app</context-root>
<security-role-mapping>
<role-name>administrator</role-name>
<principal-name>Admin</principal-name>
<group-name>administrator</group-name>
</security-role-mapping>
<security-role-mapping>
<role-name>user</role-name>
<principal-name>User1</principal-name>
<group-name>user</group-name>
</security-role-mapping>
<security-role-mapping>
<role-name>user</role-name>
<principal-name>User2</principal-name>
</security-role-mapping>
</glassfish-web-app>
```

Java EE 7 Security – Grundlagen: Sicherheitsmodelle

Man unterscheidet drei unterschiedliche Sicherheitsmodelle:

1. Deklarative Sicherheit

- ▶ Über File-Deskriptoren (web.xml, ejb-jar.xml oder proprietäre Dateien)
- ▶ Über Annotationen innerhalb der Java-Klassen

2. Programmatische Sicherheit

- ▶ Teil des Applikationscodes

3. Messagebasierte Sicherheit (nicht Teil der JEE Spezifikation)

- ▶ Mit digitalen Signaturen und Verschlüsselungen auf Message-Layer (innerhalb einer SOAP-Message [xml encryption, xml signature])

- Deklarative Sicherheit
 - Nicht alle Sicherheits-Features (z.B. andere Ressourcen ausser Servlets, Art der Authentisierung, wenn nicht der Default verwendet wird) lassen sich über Annotationen konfigurieren
 - Einstellungen im Deployment-Deskriptor überschreiben Annotationen
 - Annotationen werden nicht vererbt.
- Programmatische Sicherheit
 - Verwendung der API, welche die JSE resp JEE Plattform anbietet
- Auf die messagebasierte Sicherheit wird in diesem Kurs nicht näher eingegangen.

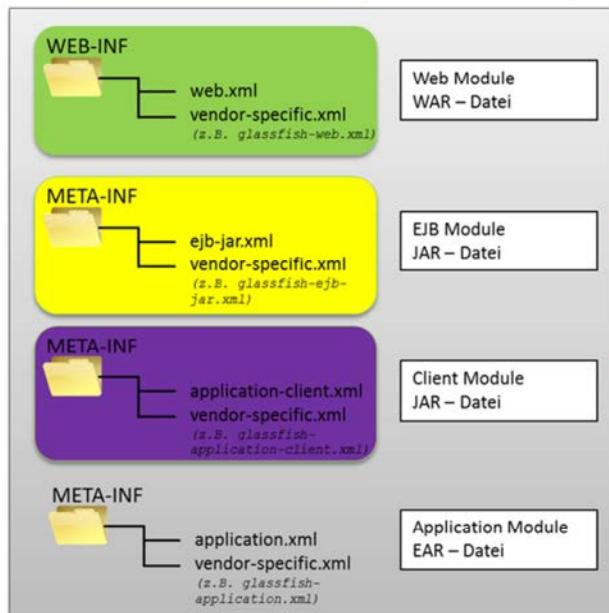
Java EE 7 Security – Grundlagen: Sicherheitsmodelle

1. Deklarative Sicherheit

- Definitionen in **Deployment-Deskriptoren** (Standard-Deskriptoren und / oder proprietäre Deskriptoren) oder
- Definitionen über **Annotationen** (JSR 250 – Security Annotations, `javax.annotation.security` siehe auch <https://jcp.org/en/jsr/detail?id=250>) innerhalb der Klassen aber ausserhalb des Programmcodes
- Vergabe von Zugriffsrechten **einzig** über Rollen (nicht über Benutzer)
- Trennung von Businesslogik und Sicherheitsaspekten
- Hintergrund: Die Implementierung der Sicherheit ist Aufgabe des JEE Containers (Web / EJB) und nicht des Programmierers

- Vorteil von Annotationen:
 - Schnell und einfach in der Anwendung
 - Transparent und leicht verständlich
- Nachteile von Annotationen:
 - Änderungen bedeuten grundsätzlich eine «Code-Änderung»
 - Es lassen sich nicht alle Einstellungen machen mit Annotationen
 - Müssen Ressourcen ausser «Java Ressourcen» geschützt werden (z.B. ein Bild) sind die Annotationen umständlicher
- Vorteile von File-Deskriptoren:
 - Übersicht (alles ist in einer Datei)
 - Keine Code-Änderungen bei Änderungen in der Sicherheit
 - Mehr Möglichkeiten
- Nachteile von File-Deskriptoren:
 - Teilweise umständlich

Java EE 7 Security – Grundlagen: Sicherheitsmodelle



- ▶ Einträge in Deployment-Deskriptoren überschreiben Einträge in Annotationen
- ▶ Proprietäre Deployment-Deskriptoren werden teilweise benötigt
- ▶ Proprietäre Deployment-Deskriptoren überschreiben Standard- Deployment-Deskriptoren (in der Regel)

Java EE 7 Security – Grundlagen: Sicherheitsmodelle

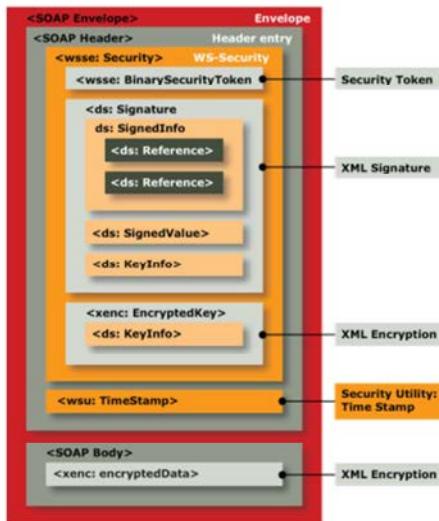
2. Programmatische Sicherheit

- Steuerung des Zugriffs **innerhalb des Programmcodes**
- Definition von Rollen in Klassen. Definition deren Berechtigungen innerhalb des Programmcodes
- **Keine Trennung** von Businesslogik (Programmcode) und Sicherheitsaspekten
- Überprüfung von Bedingungen zur Laufzeit
- Hintergrund: Deckt die deklarative Sicherheit die Anforderungen nicht oder nur teilweise ab, kann diese durch die programmatische Sicherheit ergänzt oder ersetzt werden.

- Vorteile der programmatischen Sicherheit
 - Höhere Flexibilität
 - Feinere Zugriffsberechtigungen sind möglich (z.B. auf Stufe Datensatz / Record)
- Nachteile der programmatischen Sicherheit
 - Businesslogik und Sicherheitsaspekte sind nicht mehr getrennt
 - Eine Anpassung der Sicherheit bedeutet automatisch auch eine Anpassung des Applikationscode

Java EE 7 Security – Grundlagen: Sicherheitsmodelle

3. Messagebasierte Sicherheit (Web-Service-Security)



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
37

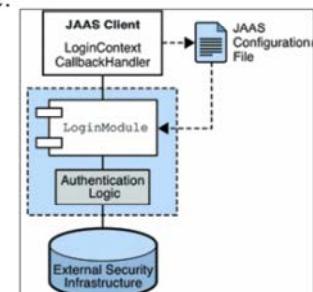
 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technik
Peter Andri

- WS-Security ist ein Standard aus dem Kontext der WS-* -Spezifikationen
- Der Standard beinhaltet Spezifikationen, die vorschreiben, wie Nachrichtenintegrität und Nachrichtenverschlüsselung im Rahmen von Webservices sichergestellt werden können. Dabei schreibt WS-Security jedoch nicht alle Details vor, sondern setzt viel mehr auf bereits bestehende Verfahren auf (XML-Signature und XML-Encryption). [siehe auch Wikipedia]
- Details zu WSS siehe auch https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

Java EE 7 Security – Grundlagen: Security Services

The Java Authentication and Authorization Service (JAAS)

- Authentisierungs-API für pluggable und sequentielle Authentisierung
- Callback-Fähigkeit zwischen Services und Applikationen
- Protokollunabhängige Verarbeitung
- Z.B. um proprietäre Authentisierungen einfach und ohne Anpassungen an Applikationen zu ermöglichen
- Notwendige Interfaces für Authentisierungsmodule:
 - `javax.security.auth.*`
- Links und Tutorial
 - <https://jcp.org/en/jsr/detail?id=115>
 - <https://docs.oracle.com/javase/7/docs/technotes/guides/security/jaas/tutorials/GeneralAcnOnly.html>
 - <https://www.youtube.com/watch?v=xv2xltxnBU>



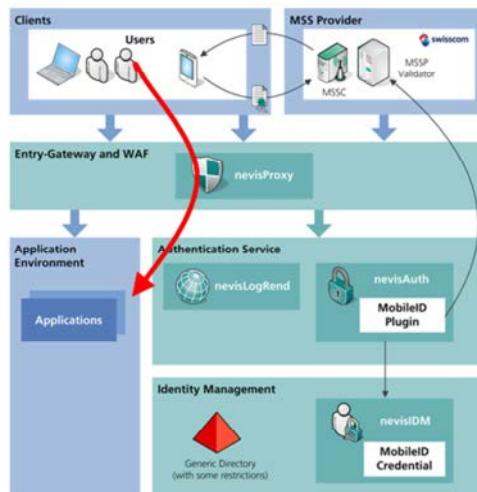
Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
38

 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Ein einfaches Beispiel eines Login-Modules kann hier gefunden werden:
 - <http://docs.oracle.com/javase/1.5.0/docs/guide/security/jaas/tutorials/SampleLoginModule.java>

Java EE 7 Security – Grundlagen: Security Services

JAAS-Login-Modul: Hier am Beispiel im Zusammenspiel mit einem Proxy



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
39



Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technik
Peter Andri

Java EE 7 Security – Grundlagen: Security Services

Java Authentication SPI for Containers (JASPI, JSR 196)

- Standard Service-Provider Interface (SPI) zur Integration eines Authentisierungsmechanismus in «message processing runtimes»
- Definition von Profilen zur Verwendung des SPI in einem spezifischen Kontext
- Notwendige Interfaces für Authentisierungsmodule:
 - `javax.security.auth.message.module.ClientAuthModule`
 - `javax.security.auth.message.module.ServerAuthModule`
- Tutorials
 - <https://docs.oracle.com/cd/E19226-01/820-7627/girp/index.html>
 - <https://docs.oracle.com/javaee/7/tutorial/doc/overview007.htm#GIRBE>
 - https://blogs.oracle.com/theaquarium/entry/an_overview_of_jaspic_1

Java EE 7 Security – Grundlagen: Security Services

Java Authorization Contract for Containers (JACC JSR-115)

- Definition eines Standardmechanismus für Authorization Provider
- Einführung neuer `java.security.Permission` - Klassen
- Erfüllung neuer JEE-Anforderung an die Autorisierung:
 - Definition von Rollen als Sammlung von Berechtigungen
 - Zuweisung von Berechtigungen zu Principal mittels Rollen
- Notwendige Interfaces :
 - `javax.security.jacc*`
- Tutorials
 - https://docs.oracle.com/javaee/7/api/javax/security/jacc/package-summary.html#package_description

- Definiert diverse Sicherheitsrichtlinien für die diversen Java-EE-Container

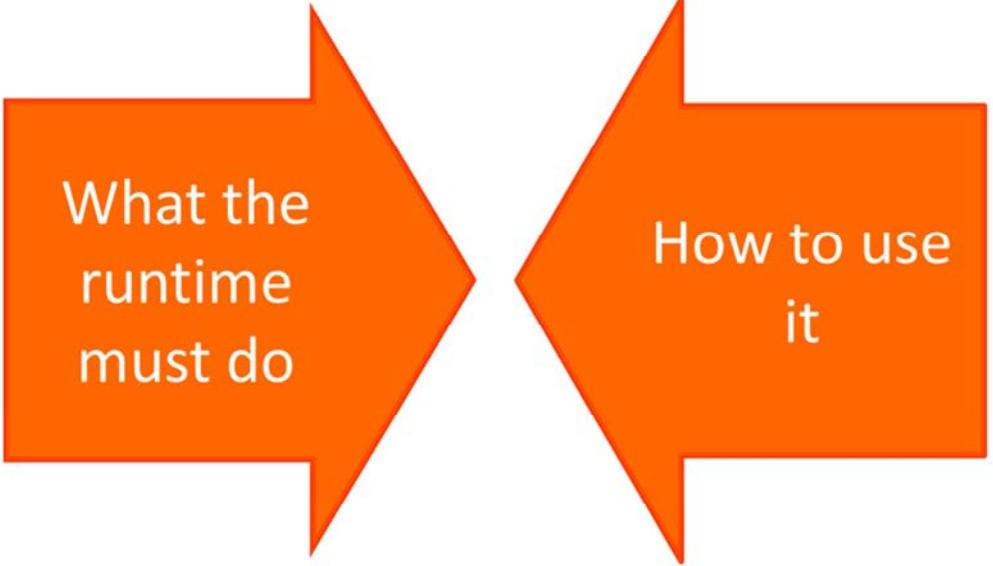
Java EE 7 Security – Grundlagen: Security Services

Java Generic Security Services (Java GSS-API)

- GSSAPI ist API für Anwendungen, die auf Security Devices zugreifen.
- GSSAPI bietet keine Sicherheit in Form eines API. Stattdessen bieten verschiedene Hersteller ihre Sicherheitssoftware in Form von Libraries an.
- Das wichtigste Feature des GSSAPI ist der Austausch von Nachrichten (Tokens), die die Implementierungsdetails vor den höheren Schichten der Anwendung verstecken.
- Links
 - <https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/BasicClientServer.html>

- Das GSSAPI selbst bietet keinerlei Sicherheit. Stattdessen bieten verschiedene Hersteller ihre Sicherheitssoftware an, oft in Form von Bibliotheken. Diese Bibliotheken präsentieren ein GSS-kompatibles Interface für Anwendungsprogrammierer, welche wiederum nur die herstellerunabhängige und standardisierte GSSAPI nutzen müssen. Wenn die Implementationen der Sicherheitsfunktionen irgendwann ausgetauscht werden müssen, bedarf es keiner Änderung an der Applikation. *[Wikipedia]*

Java EE 7 Security – Grundlagen



What the runtime must do

How to use it

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
43



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technik
Peter Andri

- Java EE Security bietet ein robustes und relativ einfach zu konfigurierendes Sicherheitskonzept für die Authentisierung, die Autorisierung und für den sicheren Datentransport.
- In Java JEE Security sind die Container (Web oder EJB) verantwortlich, diese Services zur Verfügung zu stellen. Der Entwickler muss diese nur noch nutzen.
- Der Vorteil ist: Die JEE 7 Applikation inklusive deren Sicherheits-Einstellung sind grundsätzlich portabel (kein Vendor-Lock), wenn keine proprietären Erweiterungen verwendet worden sind.

Java EE 7 Security Übung 0: Erstellen Realm

- Erstellen eines file-based Realm im Glassfish Applikationsserver mit dem Namen «SecureRealm». Dies kann im Glassfish unter «Konfigurationen->server-konfig->Sicherheit->Realms» erstellt werden.
- Der Klassename muss `<com.sun.enterprise.security.auth.realm.file.FileRealm>` heißen
- JAAS-Kontext muss `<fileRealm>` heißen.
- Das File kann unter `<${com.sun.aas.instanceRoot}/config/secure-keyfile>` abgelegt werden.
- Erstellen von zwei Gruppen `<user>` und `<administrator>`
- Erfassen von folgenden Benutzern:
 - user1 [Gruppe *user*, Password *tbd*]
 - user2 [Gruppe *user*, Password *tbd*]
 - Admin [Gruppe *administrator*, Password *tbd*]
- Aktivieren der Option `<default principal-to-role mapping>` damit der Glassfish das Role-Mapping automatisch macht

- Eine gute Anleitung zum Erstellen von Realms (z.B. JDBC Realm) kann man hier finden
 - <http://java.dzone.com/articles/jdbc-realm-and-form-based>
 - <http://www.get-the-solution.net/index-blog-1-14-166-Glassfish-4-and-Real.html>
- Hinweis: Unter «Konfigurationen->server-konfig->Sicherheit» gibt es eine Option vorhanden Namens «Zuordnung von Standard-Principal zu Rolle» (`<default principal-to-role mapping>`)
 - Diese dient dazu, dass der Applikationsserver das «Role-Mapping» automatisch macht, wenn die Rolle und Gruppenname gleich heißen. Details siehe auch «Sicherheit im Web-Tier»
- Zeitbedarf: 15 – 25 Minuten

4

Sicherheit im Web-Tier

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
46



Schweizerische Eidgenossenschaft
Confédération suisse
Confederatio Helvetica
Confederatio svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

4.1

Sicherheit im Web

Authentisierungsverfahren

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
47



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

Java EE 7 Security: Authentisierungsverfahren

- ▶ Die JEE Plattform unterstützt folgende Authentisierungsverfahren
 - ▶ **BASIC** (Basic Authentication), Default - Authentisierung
 - ▶ **FORM** (Form-Based Authentication)
 - ▶ **DIGEST** (Digest Authentication)
 - ▶ **CLIENT-CERT** (Client Certificate Authentication)
- ▶ Das Authentisierungsverfahren wird im Deploymentdeskriptor (web.xml) innerhalb des «login-config» Elementes deklariert.
- ▶ Im Element «auth-method» unterhalb des Elementes «login-config» wird das Verfahren definiert [BASIC | FORM | DIGEST | CLIENT-CERT]

```
Konfiguration eines Authentisierungsverfahrens [Form] im web.xml
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>myRealm</realm-name>
    <form-login-config>
        <form-login-page>/login/loginform.html</form-login-page>
        <form-error-page>/login/loginerror.html</form-error-page>
    </form-login-config>
</login-config>
```

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
48



Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Postfach

- Zu jeder Authentisierung wird ein Realm benötigt und angegeben. Im Element «realm-name» unterhalb des Elementes «login-config» wird der Realm definiert
- Wird kein Authentisierung-Verfahren definiert, wird das Default Authentisierungs-Verfahren (BASIC) verwendet mit dem Default Realm (im Glassfish normalerweise «file», kann aber konfiguriert werden).

Java EE 7 Security: HTTP-Status Codes

HTTP Status Codes																						
For great REST services the correct usage of the correct HTTP status code in a response is vital.																						
1xx – Informational	2xx – Successful	3xx – Redirection	4xx – Client Error	5xx – Server Error																		
<p>This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line</p> <p>100 – Continue 101 – Switching Protocols 102 – Processing</p>	<p>This class of status code indicates that the client's request was successfully received, understood, and accepted.</p> <p>200 – OK 201 – Created 202 – Accepted 203 – Non-Authoritative Information 204 – No Content 205 – Reset Content 206 – Partial Content 207 – Multi-Status</p>	<p>This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request.</p> <p>300 – Multiple Choices 301 – Moved Permanently 302 – Found 303 – See Other 304 – Not Modified 305 – Use Proxy 307 – Temporary Redirect</p>	<p>The 4xx class of status code is intended for cases in which the client seems to have erred.</p> <p>400 – Bad Request 401 – Unauthorized 402 – Payment Required 403 – Forbidden 404 – Not Found 405 – Method Not Allowed 406 – Not Acceptable 407 – Proxy Authentication Required 408 – Request Timeout 409 – Conflict 410 – Gone 411 – Length Required 412 – Precondition Failed 413 – Request Entity Too Large 414 – Request URI Too Long 415 – Unsupported Media Type 416 – Requested Range Not Satisfiable 417 – Expectation Failed 422 – Unprocessable Entity 423 – Locked 424 – Failed Dependency 425 – Unordered Collection 426 – Upgrade Required</p>	<p>Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request.</p> <p>500 – Internal Server Error 501 – Not Implemented 502 – Bad Gateway 503 – Service Unavailable 504 – Gateway Timeout 505 – HTTP Version Not Supported 506 – Variant Also Negotiates 507 – Insufficient Storage 508 – Not Extended</p>																		
<p>Examples of using HTTP Status Codes in REST</p> <p>201 – When doing a POST to create a new resource it is best to return 201 and not 200. 204 – When deleting a resources it is best to return 204, which indicates it succeeded but there is no body to return. 301 – If a 301 is returned the client should update any cached URI's to point to the new URI. 302 – This is often used for temporary redirect's, however 303 and 307 are better choices. 409 – This provides a great way to deal with conflicts caused by multiple updates. 501 – This implies that the feature will be implemented in the future.</p> <p>Special Cases</p> <p>306 – This status code is no longer used. It used to be for switch proxy. 418 – This status code from RFC 2324. However RFC 2324 was submitted as an April Fools' Joke. The message is <i>I am a teapot</i>.</p>																						
<table border="1"> <thead> <tr> <th>Key</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Black</td> <td>HTTP version 1.0</td> </tr> <tr> <td>Blue</td> <td>HTTP version 1.1</td> </tr> <tr> <td>Aqua</td> <td>Extension RFC 2295</td> </tr> <tr> <td>Green</td> <td>Extension RFC 2518</td> </tr> <tr> <td>Yellow</td> <td>Extension RFC 2778</td> </tr> <tr> <td>Orange</td> <td>Extension RFC 2817</td> </tr> <tr> <td>Purple</td> <td>Extension RFC 3648</td> </tr> <tr> <td>Red</td> <td>Extension RFC 4918</td> </tr> </tbody> </table>					Key	Description	Black	HTTP version 1.0	Blue	HTTP version 1.1	Aqua	Extension RFC 2295	Green	Extension RFC 2518	Yellow	Extension RFC 2778	Orange	Extension RFC 2817	Purple	Extension RFC 3648	Red	Extension RFC 4918
Key	Description																					
Black	HTTP version 1.0																					
Blue	HTTP version 1.1																					
Aqua	Extension RFC 2295																					
Green	Extension RFC 2518																					
Yellow	Extension RFC 2778																					
Orange	Extension RFC 2817																					
Purple	Extension RFC 3648																					
Red	Extension RFC 4918																					

Quelle <http://www.sadev.co.za/content/http-status-codes-cheat-sheet>

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
49

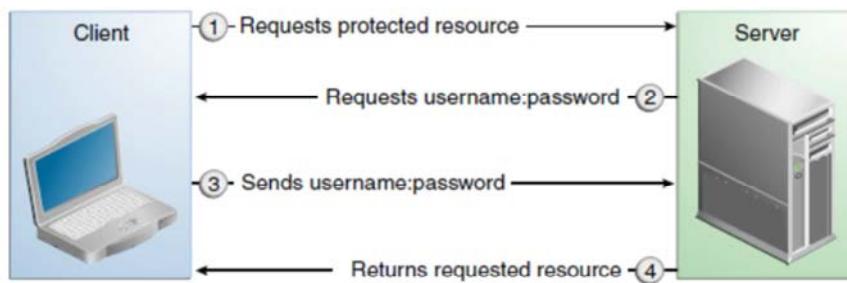
 Schweizerische Eidgenossenschaft
Confédération suisse
Confederatio Helvetica
Confederatio svitza

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

- Eine kleine HTTP-Status Tabelle zum Nachschlagen. In unserem Kontext wichtig ist der
 - HTTP Status 401 (Unauthorized)
 - HTTP Status 403 (Forbidden)
 - HTTP Status 302 (Found)
 - Und natürlich HTTP Status Code 200 (OK)

Java EE 7 Security: Authentisierungsverfahren BASIC

HTTP BASIC Authentication: Übersicht

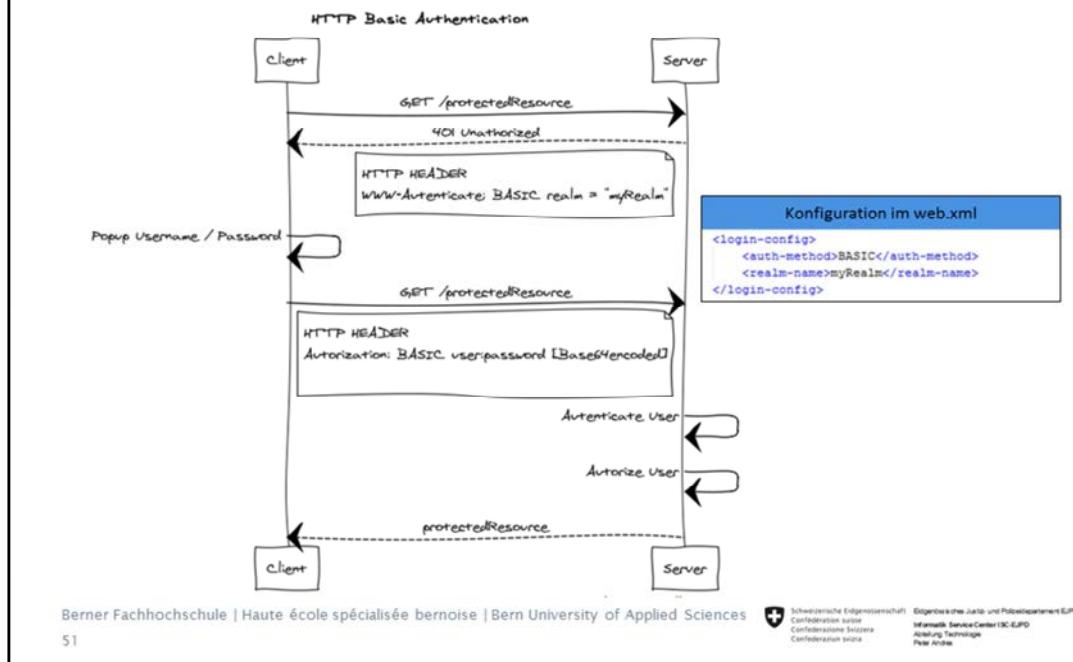


Quelle: Oracle JEE 7 Tutorial

- Die Basic Authentication ist **RFC 2617** spezifiziert

Java EE 7 Security: Authentisierungsverfahren BASIC

HTTP BASIC Authentication: Detail

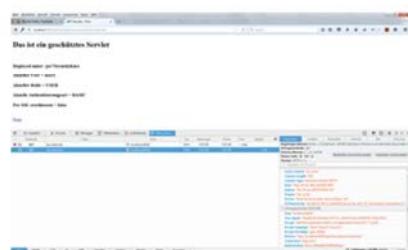


- Die Basic Authentication nach **RFC 2617** ist die häufigste Art der HTTP-Authentisierung.
- Im Deployment-Deskriptor (web.xml), im Element «auth-method» wird BASIC angegeben. Weiter ist im Element «realm-name» der zu verwendende Realm anzugeben.
- Der Webserver fordert mit Statuscode 401 (unauthorized) und dem HTTP-Header «WWW-Authenticate: Basic realm=myRealm» eine Authentisierung an, nachdem der Client auf eine geschützte Ressource zugreifen will.
- Der Webclient fragt nach Username / Passwort (Browser-Popup) und sendet die Authentisierungs-Daten in Authorization-Header «Authorization» in der Form «Benutzername:Passwort» Base64-codiert an den Server.
- Zum Beispiel als: `Authorization Basic dXNlcjpQQVNTV09SVA==`
`Basic user:PASSWORT` (Decodiert)
- Ohne zusätzliche Massnahmen ist das ein unsicheres Verfahren (Username und Passwörter gehen im Klartext über das Netzwerk)
- Der Webserver wird (ohne zusätzliche Massnahmen) vom Client nicht authentisiert

Java EE 7 Security: Authentisierungsverfahren BASIC

In der Praxis sieht das dann folgendermassen aus ...

DEMO



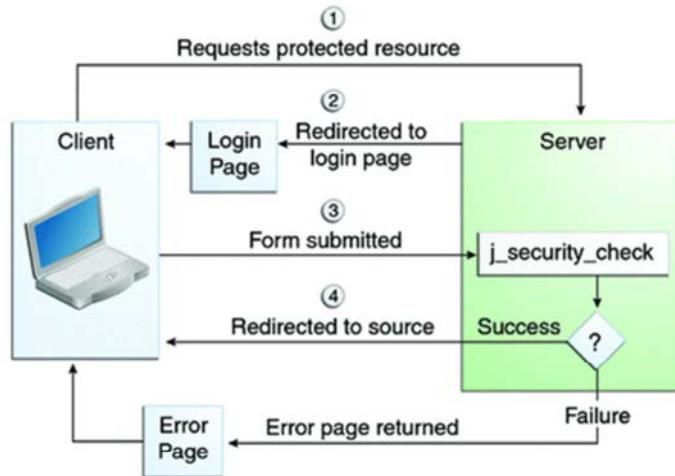
Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
52



Eidgenössisches Justiz- und Polizeidepartement EJPD
Informatik Service-Center ISC-EJPD
Abteilung Technologie
Peter Andree

- Das einfache Beispiel kann unter «JEE7_Security_Kurs_Sample.zip» in NetBeans importiert und getestet werden
 - Dazu muss im Glassfish Server unter «server-config/Sicherheit/Realms» ein neuer Realm namens «SecureRealm» konfiguriert werden (! JAAS-Kontext = fileRealm)
 - Anschliessend muss ein User in der Gruppe USER angelegt werden
 - Analyse des Web-Traffic mit Firefox, dabei die Entwicklungstools (Netzwerkanalyse) einschalten
 - Decodieren des Base64-Headers kann unter «<https://www.base64decode.org/>» gemacht werden. Vorsicht – niemals produktive Daten und oder Passwörter online decodieren.

Java EE 7 Security: Authentisierungsverfahren FORM



Quelle: Oracle JEE 7 Tutorial

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
53

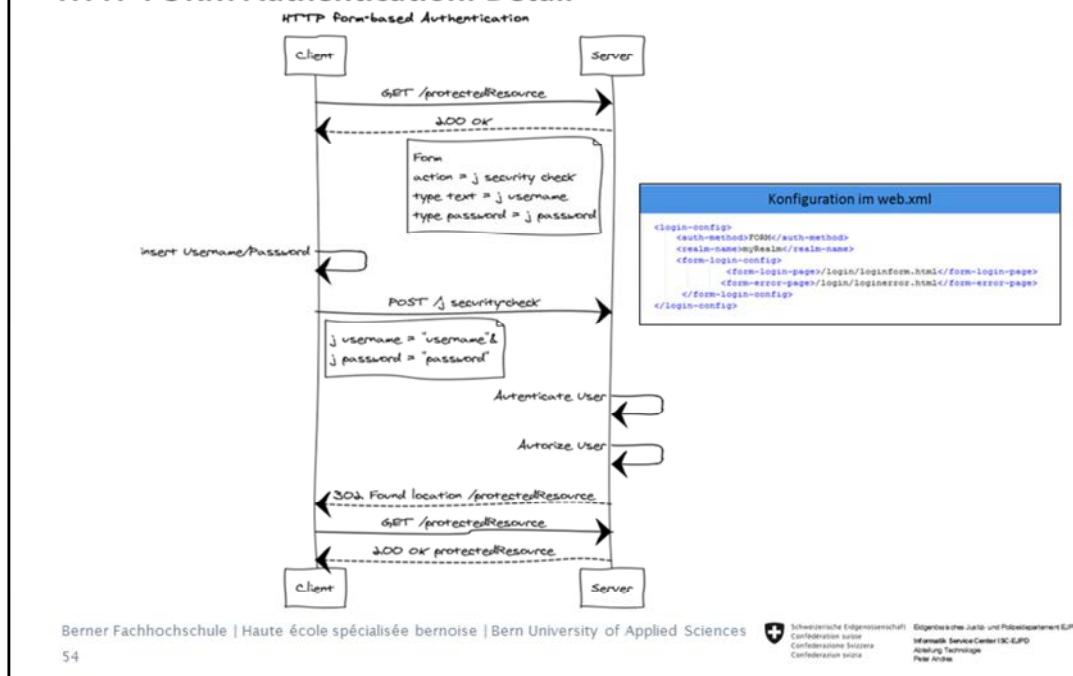


Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Die form-based Authentication ist ebenfalls eine häufige Art der HTTP-Authentisierung (jedoch kein RFC !).

Java EE 7 Security: Authentisierungsverfahren FORM

HTTP FORM Authentication: Detail



- Für formbased Login gibt es keinen RFC.
- Im Deployment-Deskriptor (web.xml), im Element «auth-method» wird FORM angegeben. Weiter ist im Element «realm-name» der zu verwendende Realm anzugeben.
- Das Formular, resp die Loginpage und eine Errorpage (im Fehlerfall) kann im Element «form-login-config» Element definiert werden. Diese Seiten / Formulare kann der Entwickler selber gestalten.
- Der Webserver gibt dem Client ein Formular für die Authentisierung mit:
 - Action = j_security_check
 - Type text = j_username
 - Type password = j_password
- Die Action, das Inputfeld «Text» und das Passwortfeld «Passwort» muss zwingend j_security_check, j_username und j_password heißen. Dies muss per HTTP POST (Action = j_security_check) dem Webserver übermittelt werden. Nur in diesem Fall wird die Authentisierung automatisch durch den Webserver gemacht

- Der Webclient «posted» den Username und das Passwort per HTTP POST (im Klartext).
- Der Server liefert die Ressource aus nach erfolgreicher Authentisierung und Autorisierung.
- Ohne zusätzliche Massnahmen ein unsicheres Verfahren (Username und Passwörter gehen im Klartext über das Netzwerk)
- Der Webserver wird (ohne zusätzliche Massnahmen) vom Client nicht authentisiert

Java EE 7 Security: Authentisierungsverfahren Form

In der Praxis sieht das dann folgendermassen aus ...



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
55

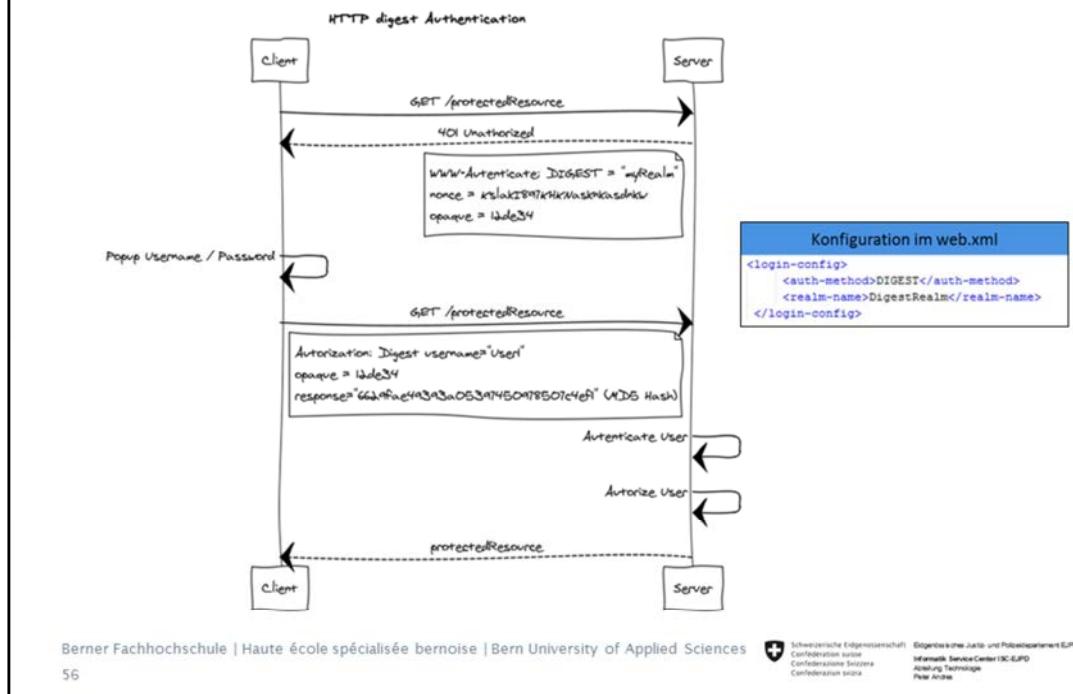
 Schweizerische Eidgenossen
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Postfach

- Das einfache Beispiel kann unter «JEE7_Security_Kurs_Sample.zip» in NetBeans importiert und getestet werden
 - Dazu muss im Glassfish Server unter «server-config/Sicherheit/Realms» ein neuer Realm namens «SecureRealm» konfiguriert werden (! JAAS-Kontext = fileRealm)
 - Anschliessend muss ein User in der Gruppe USER angelegt werden
 - Die Login- und Errorseite kann individuell gestaltet werden.

Java EE 7 Security: Authentisierungsverfahren DIGEST

HTTP DIGEST Authentication: Detail



- Die DIGEST Authentisierung nach **RFC 2617** (analog BASIC Authentication) wird eher selten verwendet.
- Funktionsweise im Grundsatz wie BASIC Authentication – jedoch wird anstelle des Plaintext-Passwort (resp. anstelle der Base64-Authentisierungsdaten) eine Hash (digitaler Fingerabdruck) als Authentisierungsdaten versendet.
- Im Beispiel oben wurden nicht alle HTTP Headers aufgelistet, welche zwischen Client und Server hin und her geschickt werden. Details siehe auch RFC oder folgende Folien.
- Der Benutzer (Username) und das Passwort werden anstelle Base64-Header als MD5 Digest (Hash) versendet:
 - $$\text{MD5}(\text{username} + ":" + \text{REALM} + ":" + \text{password}) + ":" + \text{NONCE} + ":" + \text{MD5}(\text{http-method} + ":" + \text{uri}))$$
- Auch andere Algorithmen und Verfahren sind möglich (siehe RFC 2617 <https://www.ietf.org/rfc/rfc2617.txt>)

Java EE 7 Security: Authentisierungsverfahren DIGEST

Parameter Authentisierungs-Request Webserver -> Webclient

Attribut	Beschreibung
algorithm (optional)	Der zu verwendende Algorithmus. Wenn die Angabe fehlt, wird MD5 angenommen.
domain (optional)	Koma-separierte Liste von URIs
nonce	Ein Zufalls-String, der serverseitig erzeugt wird. Dieser String wird dann clientseitig zusammen mit dem Passwort benutzt, um daraus den Digest zu bestimmen. Wird nur einmal verwendet.
opaque (optional)	Ein Zufalls-String, der serverseitig erzeugt wird. Dieser muss unverändert an den Server zurückgeschickt werden.
qop	Zeigt die Qualität an. (Quality of protection) [auth" oder "auth-int"]
realm	Der Name einer geschützten Security-Domäne.
state	Ein boolesches Attribut. Der Wert true bedeutet, dass der Wert von nonce veraltet ist und der Webclient nochmal ein Request machen muss.

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
57



Eidgenössische Justiz und Polizeidepartement EJPD
Informatik Service Center ITC-EJPD
Abteilung Technologie
Peter Andri

- Beispiel Anfrage (Request)

```
GET /protected/test.html HTTP/1.1
```

```
WWW-Authenticate: Digest
    realm=<user@test.com>,
    domain=<mydomain>,
    nonce="dcd98b7102dd2f0e8b11d0f600bfb0c0",
    opaque="5ccc069c403ebaf9f0171e9517f40e41",
    qop="auth,auth-int",
    stale="false",
    algorithm="MD5"
```

- Details zu den einzelnen Werten siehe auch rfc 2617
<https://www.ietf.org/rfc/rfc2617.txt>

Java EE 7 Security: Authentisierungsverfahren DIGEST

Parameter Authentisierungs-Request Webclient -> Webserver

Attribut	Beschreibung
qop	Quality of protection, welche der Client angewendet hat [auth" oder "auth-int"]
cnonce	Nonce vom Client
nc	Nonce Count (vermeiden mehrfachem senden von Requests)
response	32 hex String mit Username und Passwort (gehashed)
uri	Der URI der Anfrage
opaque	Ein Zufalls-String, der serverseitig erzeugt wird. Dieser muss unverändert an den Server zurückgeschickt werden.
algorithm	Der zu verwendende Algorithmus. Wenn die Angabe fehlt, wird MD5 angenommen.

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
58



Eidgenössisches Justiz- und Polizeidepartement (OJ-EPD)
Informatik-Service-Center (ISC-EPD)
Abteilung Technologie
Peter Andree

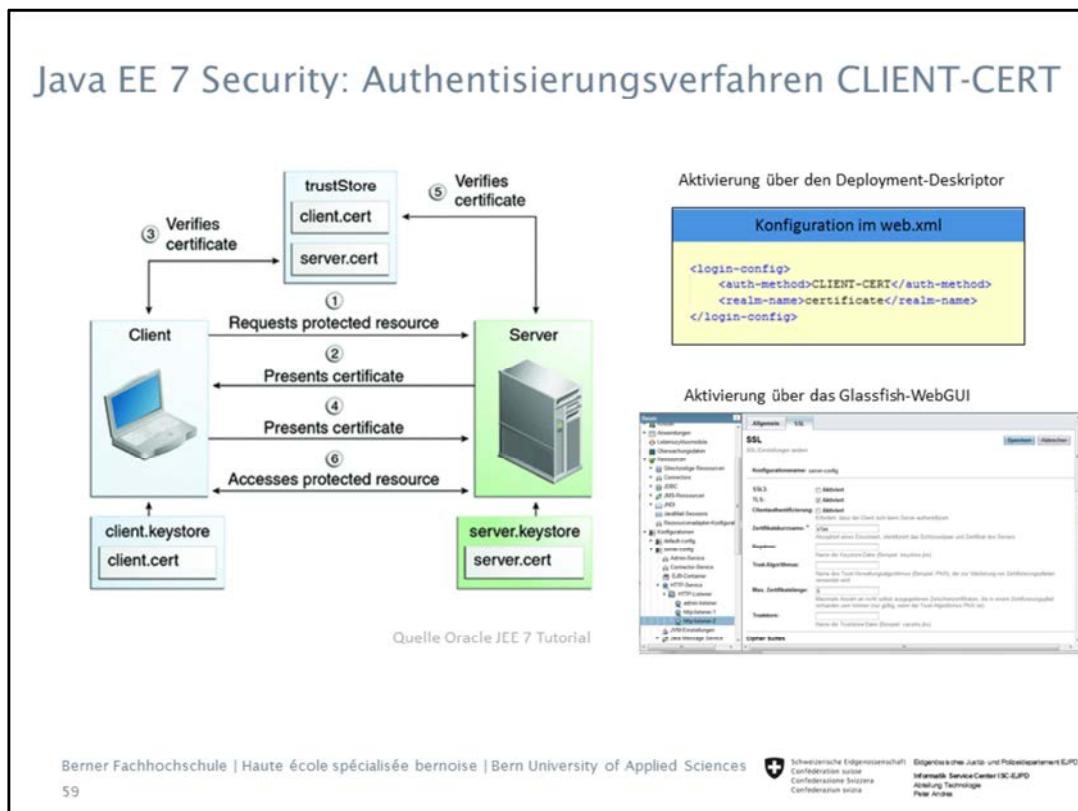
- Beispiel Antwort (Response)

```
GET /protected/test.html HTTP/1.1
```

```
Authorization: Digest username=<User>,
realm=<SecureRealm>,
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c0",
uri="/dir/index.html",
qop=auth,
nc=00000001,
cnonce="0a4f113b",
response="6629fae49393a05397450978507c4ef1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
algorithm="MD5"
```

- Details zu den einzelnen Werten siehe auch rfc 2617
<https://www.ietf.org/rfc/rfc2617.txt>

Java EE 7 Security: Authentisierungsverfahren CLIENT-CERT

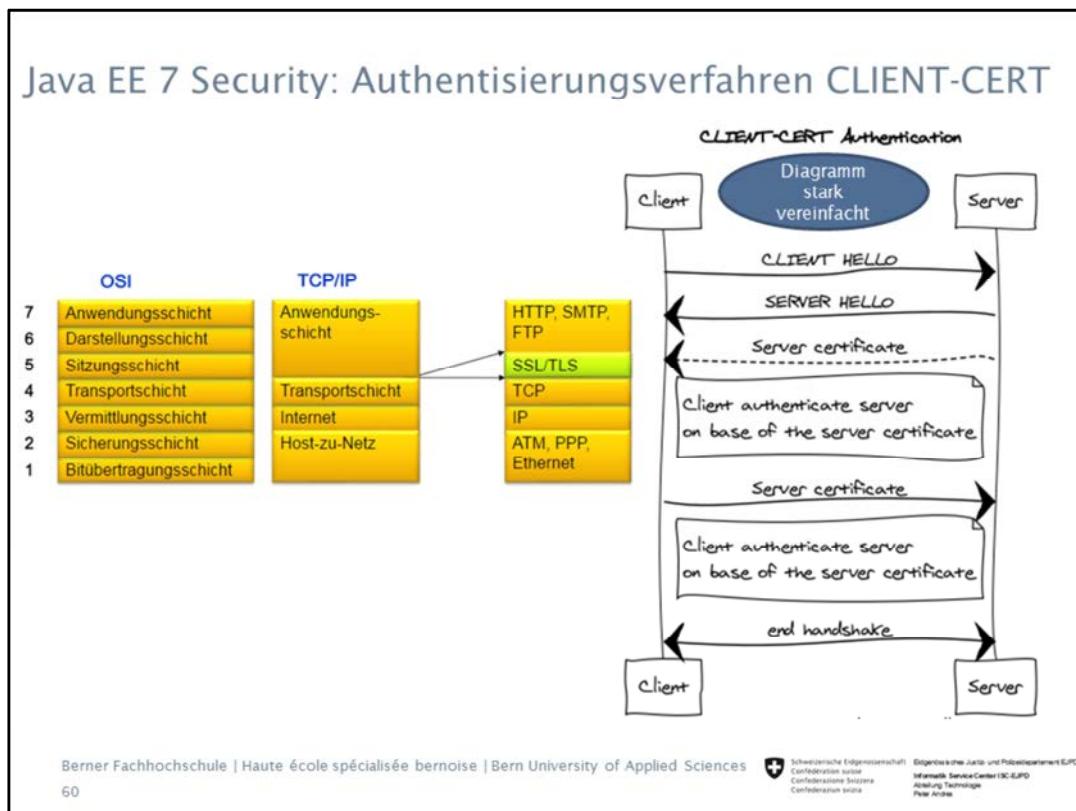


Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
59

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun Svizra
Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technik
Postfach

- Details siehe auch Transportsicherheit
- Der Server wird vom Client anhand des Server-Zertifikates authentisiert
- Der Client wird vom Server anhand des Client-Zertifikates authentisiert
- Der Subject Common Name des X509V3 Clientzertifikates enthält den Principal-Name
- Die Sicherheit (insbesondere wenn die Client-Zertifikate auf einem Hard-Token [z.B. Smartcard] liegen) ist deutlich höher als bei der vorhergehenden Verfahren. In diesem Fall (bei einem Hardtoken) spricht man von einer starken, 2 Faktor Authentisierung (1. Ich weiss etwas = Pin zur Smartcard, 2. ich besitze etwas = Smartcard)
- Details siehe auch Kapitel «Transportsicherheit»

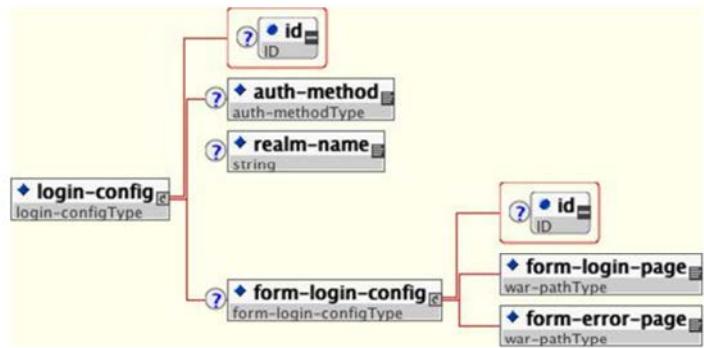
Java EE 7 Security: Authentisierungsverfahren CLIENT-CERT



- Diese Authentisierung passiert auf OSI Layer 4/5 (und nicht wie in den anderen Authentisierungsmethoden auf OSI Layer 7)
- Der Webserver wird anhand des Serverzertifikates authentisiert. Vertraut man dem Serverzertifikat, wird auch dem Server vertraut.
Das ist das einzige Verfahren, wo neben dem Client auch der Server authentisiert wird. (ohne die Verwendung von Transportsicherheit)
- Der Client (die Person) wird anhand des Clientzertifikates authentisiert

Java EE 7 Security: Authentisierungsverfahren

Übersicht des Schemas / Elementes (*login-config*) im *web.xml*



Java EE 7 Security: Authentisierungsverfahren

Ein Sicherheits-Vergleich der Authentisierungeverfahren

Methode	Sicherheits-niveau	Aufwand für Implementierung	Server-anforderungen	Kommentare
Standard-Authentisierung	Niedrig	Niedrig	Benutzerverwaltung	Authentisierungsinformationen und Daten werden unverschlüsselt übertragen!
Formularbasierte Authentisierung ohne gesicherte Übertragung	Niedrig	Niedrig bis mittel	Implementierung in der jeweiligen Anwendung	Authentisierungsinformationen und Daten werden unverschlüsselt übertragen!
Digest-Authentisierung	Mittel	Niedrig	Benutzerverwaltung	Daten werden unverschlüsselt übertragen.
Formularbasierte Authentisierung über SSL	Hoch	Mittel bis hoch	SSL-Unterstützung im Server, Implementierung in der jeweiligen Anwendung	Authentisierungsinformationen und Daten werden verschlüsselt übertragen!
Zertifikatbasierte Authentisierung über SSL	Hoch bis sehr hoch	Hoch bis sehr hoch	Installation von Server-Zertifikaten, Zertifikatsverwaltung, Public-Key Infrastruktur.	Wird hauptsächlich für sichere Transaktionen über das Internet verwendet.

Quelle BSI

- Die Sicherheits-Einstufung der Authentisierungsverfahren nach BSI – siehe auch https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m04/m04176.html
- Grundsätzlich sind die Authentisierungs-Methoden ohne Transportsicherheit (SSL) **unsicher!** – dies mit Ausnahme meldungsbasierter Sicherheit, welche auf digitalen Signaturen und kryptografischen Verschlüsselungsverfahren basiert.

4.2

Sicherheit im Web

Deklarative Sicherheit

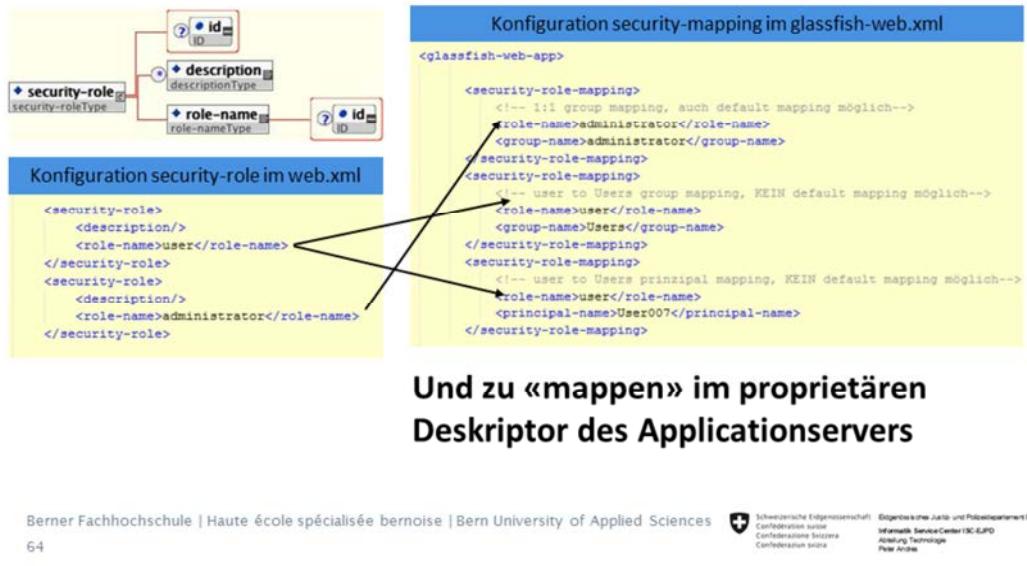
Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
63



Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Alle verwendeten Rollen sind zu deklarieren im `web.xml` Deskriptor
Innerhalb des Elements `«security-role»`.



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
64

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement EJPD
Informatik Service Center ISC-EJPD
Abteilung Technik
Peter Andri

- Als erster Punkt, müssen die Rollen der Webapplikation (`web.xml`) im Element `«security-role»` definiert gemacht werden.
- Anschliessend muss das Mapping (Rolle / Gruppe) resp. (Rolle / Prinzipal) definiert werden (proprietär)
- Die Namen der Rollen im `web.xml` müssen identisch sein mit den Namen in der Mapping-Datei (`glassfish-web.xml`)
- Der Glassfish Application Server bietet auch ein automatisches Mapping an – dies wenn die Namen der Gruppen und Rollen identisch sind (proprietär Glassfish, beschrieben unter `«default principal-to-role mapping»`):
 - *The role names used in the application are often the same as the group names defined on the GlassFish Server. Under these circumstances, you can enable a default principal-to-role mapping on the GlassFish Server by using the Administration Console. The task To Set Up Your System for Running the Security Examples explains how to do this. All the tutorial security examples use default principal-to-role mapping.*
- Struktur des `web.xml` resp `glassfish-web.xml` siehe auch
<https://docs.oracle.com/cd/E19226-01/820-7693/beaxg/index.html>

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Im `web.xml` können (optional) im Element `<security-role-ref>` Rollen verlinkt werden

Konfiguration security-role-ref im `web.xml`

```
<servlet>
  <security-role-ref>
    <role-name>admin</role-name>
    <role-link>administrator</role-link>
  </security-role-ref>
  <description>Logout Servlet</description>
  <servlet-name>LogoutServlet</servlet-name>
  <servlet-class>ch.bfh.web.LogoutServlet</servlet-class>
</servlet>

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

- Diese zusätzliche Abstraktion wird eher selten verwendet und ist optional.
- Diese Verlinkung bietet eine zusätzliche Administration. Dies kann z.B. verwendet werden, wenn zur Entwicklungszeit die Rollennamen noch nicht bekannt sind. Dann muss der Code nicht angepasst werden, wenn sich die Rollennamen ändern
- Im Servlet `«LogoutServlet»` kann z.B. die Rolle `«admin»` anstelle der Rolle `«administrator»` verwendet werden – z.B. wird die Methode
 - `isUserInRole('admin')` für einen User in der Rolle `«administrator»` `true` zurückliefern
- Das Element `«<role-link>»` muss auf eine bereits deklarierte Rolle im Element `«<security-role>»` zeigen

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Absichern (*security-constraint*) von Ressourcen via Deskriptor (*web.xml*)



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
66

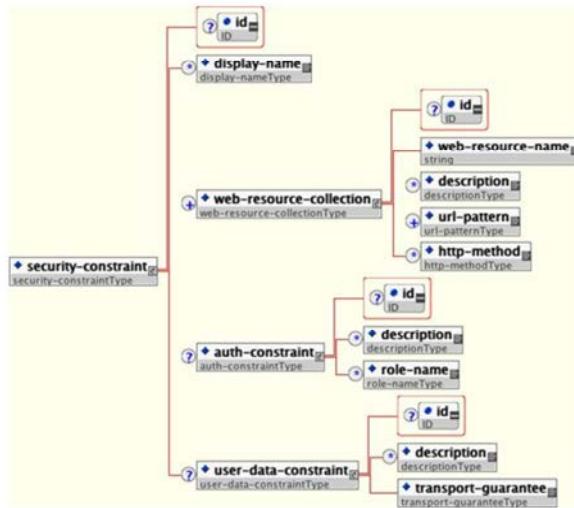
 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement EJPD
Informatik Service Center ISC-EJPD
Abteilung Technik
Peter Andri

- Eine «security constraint» definiert die Zugriffsberechtigung für ein Set von Ressourcen für bestimmte Methoden
- Pro URL und Applikation kann eine oder mehrere «security constraint» erzeugt werden. Pro «security constraint» können 1-n «web-resource-collection» definiert werden.
- Im Element «url-pattern», kann angegeben werden, welche Ressourcen geschützt werden sollen. Im Beispiel werden **alle** Ressourcen **unterhalb** /protected/ (nach dem Root Context) geschützt. Regex und Wildchars innerhalb des Pattern sind nicht erlaubt z.B. /protected/*/secure/*.
Details siehe auch
http://download.oracle.com/otndocs/jcp/servlet-3_1-fr-eval-spec/index.html «Kapitel 12.2 Specification of Mappings»
Ein Pattern wie «/*» nennt man Pfad-Mapping
Ein Pattern wie «*.» nennt man Extension Mapping
- Wenn kein Element «http-method» vorhanden ist, werden automatisch alle HTTP Methoden geschützt.
- Werden eines oder mehrere «http-method» Elemente angegeben, werden genau diese HTTP Methoden geschützt (die anderen nicht). So wäre im Beispiel die HTTP Methode «HEAD» nicht geschützt.

- Mit dem Element «http-method-omission» können HTTP-Methoden ausgeschlossen werden vom Schutz. (Im Beispiel nicht angewendet)
- Tipp: Grundsätzlich keine Angabe, damit alle HTTP-Methoden geschützt sind !
- Innerhalb des Elementes «auth-constraint» können die Rollen (0-n) definiert werden, welche auf die Ressource gemäss «url-pattern» Zugriff haben.
- Ist das Element «auth-constraint» leer, wird kein Zugriff auf die Ressource gewährleistet.
- Fehlt das «auth-constraint» Element ganz, wird der User nicht authentisiert!
- Jede Rolle im Element «role-name» muss im web.xml deklariert sein (im Element «security-role»)
- Die Namen der Rollen sind case-sensitiv.
- Die Rollen in der Applikation muss der Container auf User / Gruppen mappen können. Dazu braucht es ein proprietäres Mapping, meist ein xml File (beim Glassfish «glassfish-web.xml»), oder das «default principal-to-role mapping» muss eingeschaltet sein
- Das Element «transport-guarantee» definiert, ob der Datentransport verschlüsselt erfolgen muss oder nicht
- Bei der Transportsicherheit (im allgemeinen SSL) kann gewählt werden zwischen NONE, CONFIDENTIAL und INTEGRAL, wobei CONFIDENTIAL und INTEGRAL von den JEE Applikations-Server identisch behandelt werden. Mehr dazu im Kapitel «Transportsicherheit».

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Übersicht des Schemas (*security-constraint*) im *web.xml*



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
67

 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Beschreibung der Sub-Elemente des `<security-constraint>` im `web.xml`

`<security-constraint>` (Teil 1)

Subelement	Konten	Beschreibung
<code><web-resource-collection></code>	<code><web-resource-name></code>	Name der Ressource
	<code><description></code>	Beschreibung
	<code><url-pattern></code>	Definition, welche URL (Resoure) geschützt werden soll <ul style="list-style-type: none">• Keine Wildcards außer * erlaubt• /path/* bedeutet alle Ressourcen unterhalb des entsprechenden Pfades [path-mapping]• *.htm bedeutet alle htm Ressourcen [extension mapping]• Ein leerer String ("") ist ein spezielles URL Pattern, dass genau auf den Root-Kontext passt. In diesem Fall ist die Path Info "/"• Alle anderen url-patterns (Strings) müssen genau auf die Ressourcen passen

- A mapping that contains the pattern `<url-pattern>/</url-pattern>` matches a request if no other pattern matches. This is the *default mapping*. The servlet mapped to this pattern is called the *default servlet*.

The default mapping is often directed to the first page of an application. Explicitly providing a default mapping also ensures that malformed URL requests into the application return are handled by the application rather than returning an error. The servlet-mapping element below maps the Welcome servlet instance to the default mapping.

- `<servlet-mapping> <servlet-name>Welcome</servlet-name> <url-pattern>/</url-pattern> </servlet-mapping>`
For the context that contains this element, any request that is not handled by another mapping is forwarded to the Welcome servlet.

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Beschreibung der Sub-Elemente des `<security-constraint>` im `web.xml`

`<security-constraint>` (Teil 2)

Subelement	Konten	Beschreibung
<code><web-resource-collection></code>	<code><http-method></code>	<p>Definition, welche HTTP Methode in der constraint geschützt werden soll</p> <ul style="list-style-type: none">Keine Angabe bedeutet, dass alle HTTP-Methoden durch die constraint geschützt sind.Explizit angegebene HTTP-Methoden bedeuten, dass genau die angegebenen Methoden durch die constraint geschützt sind. Alle anderen Methoden sind ungeschützt
	<code><http-method-omission></code>	Definition, welche HTTP Methode in der constraint vom Schutz ausgeschlossen werden soll

Tipp:

- Keine HTTP-Methoden in der constraint angeben (weglassen des Elements `<http-method>`, d.h alle HTTP-Methoden sind geschützt)
- Sollen HTTP-Methoden in der constraint ausgenommen werden, ist bevorzugt das Element `<http-method-omission>` zu verwenden.

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Beschreibung der Sub-Elemente des `<security-constraint>` im `web.xml`

`<security-constraint>` (Teil 3)

Subelement	Konten	Beschreibung
<code><auth-constraint></code>	<code><description></code>	Beschreibung
	<code><role-name></code>	<p>Name der Rolle, welche Zugriff auf die im Element <code><web-resource-collection></code> definierte Ressource haben soll.</p> <ul style="list-style-type: none">• Ist das <code><auth-constraint></code> Element leer, wird kein Zugriff auf das in der constraint definierte Ressource gewährt• Fehlt das <code><auth-constraint></code> Element ganz, wird keine Authentisierung / Autorisierung durchgeführt• <code><role-name>*</role-name></code> ist eine Definition dass alle Rollen Zugriff haben (welche in <code><security-role></code> definiert wurden)• <code><role-name>**</role-name></code> ist eine Definition dass alle User (unabhängig der Rolle) Zugriff haben• Jede Rolle im <code><auth-constraint></code> Element muss auch im Element <code><security-role></code> definiert sein

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Beschreibung der Sub-Elemente des `<security-constraint>` im `web.xml`

`<security-constraint>` (Teil 4)

Subelement	Konten	Beschreibung
<code><user-data-constraint></code>	<code><description></code>	Beschreibung
	<code><transport-guarantee></code>	<p>Definition wie resp ob eine Transportverschlüsselung aktiviert werden soll oder nicht</p> <ul style="list-style-type: none">• Werte sind NONE (keine Verschlüsselung), CONFIDENTIAL (Verschlüsselung) oder INTEGRAL (Verschlüsselung)• Es gibt keinen Unterschied zwischen CONFIDENTIAL und INTEGRAL (INTEGRAL wird nicht von allen Servern unterstützt)• CONFIDENTIAL oder INTEGRAL sagt nichts über die Stärke (TLS Version, Chiphers) der Verschlüsselung aus• Fehlt das <code><user-data-constraint></code> Element ganz, wird keine Verschlüsselung für das entsprechende constraint forciert.• Das Element <code><user-data-constraint></code> darf nicht leer sein.

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Weitere sicherheitsrelevante Elemente im Deskriptor (web.xml)

```
<session-config>
  <session-timeout>30</session-timeout>
</session-config>

<servlet>
  <servlet-name>SecureServlet</servlet-name>
  <security-role-ref>
    <role-name>TheServletRole</role-name>
    <role-link>TheApplicationRole</role-link>
  </security-role-ref>
  <servlet-class>ch.admin.jeeKurs.SecureServlet</servlet-class>
  //use this role for outgoing call's
  <run-as>administrator</run-as>
</servlet>
```

Definiert das Session-Timeout der Applikation

Alias einer Rolle, welche innerhalb des Servlets verwendet werden kann

Das «<run-as>» Element spezifiziert, mit welcher Rolle der Benutzer vom Servlet **ausgehende** Calls ausführt.

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

```
package javax.servlet.annotation;  
@Inherited  
@Documented  
@Target(value=TYPE)  
@Retention(value=RUNTIME)  
  
public @interface ServletSecurity {  
    HttpConstraint value();  
    HttpMethodConstraint[] httpMethodConstraints()  
}
```

Annotation	Attribute	Wert / Beschreibung
@ServletSecurity	@HttpConstraint	Constraint, welche auf alle HTTP Methoden wirkt
	@HttpMethodConstraint	Constraint, welche auf eine spezifische HTTP Methoden wirkt

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Absichern (*security-constraint*) von Ressourcen via Annotationen (Servlet)

Listing: Annotation in der Klasse SecureDecServlet.java

```

@WebServlet(name = "SecureDecServlet", urlPatterns = {"/SecureDecServlet/*"})
@ServletSecurity(httpMethodConstraints = {
    @HttpMethodConstraint(
        /*Denying access to all HTTP GET methods
        (all other HTTP methods are allowed)*/
        value = "GET",
        /*default authorization semantic, when no roles specified
        by the array rolesAllowed*/
        emptyRoleSemantic = EmptyRoleSemantic.DENY,
        /*Specifying that the connection requires no encryption
        transportGuarantee = ServletSecurity.TransportGuarantee.NONE,
        /*Requiring that users must have membership in role "user"
        (for HTTP GET methods)*/
        rolesAllowed = {"user"}),
    @HttpMethodConstraint(
        /*Denying access to all HTTP POST methods */
        value = "POST",
        /*Specifying that the connection needs encryption
        transportGuarantee = TransportGuarantee.CONFIDENTIAL,
        /*Requiring that users must have membership in role "admin"
        (for HTTP POST methods)*/
        rolesAllowed = {"admin"})
})

```

The diagram illustrates the annotations from the code and their corresponding security concepts:

- Geschützte Ressourcen**: Points to the line `@WebServlet(name = "SecureDecServlet", urlPatterns = {"/SecureDecServlet/*"})`.
- HTTP Methoden, welche geschützt werden sollen**: Points to the `@HttpMethodConstraint` annotations for GET and POST methods.
- Verhalten, wenn keine Rolle definiert wird in «rolesAllowed»**: Points to the `emptyRoleSemantic = EmptyRoleSemantic.DENY` setting.
- Definition der Transport-Sicherheit**: Points to the `transportGuarantee = ServletSecurity.TransportGuarantee.NONE` and `transportGuarantee = TransportGuarantee.CONFIDENTIAL` settings.
- Rollen, welche für diese Ressource zugelassen sind**: Points to the `rolesAllowed = {"user"}` and `rolesAllowed = {"admin"}` settings.
- Zweite Constraint**: Points to the second `@HttpMethodConstraint` block for the POST method.

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
74

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement (EJPO)
Informatik Service Center (ISC-EJPO)
Abteilung Technologie
Peter Andri

- Grundsätzlich können hier im Servlet dieselben Constraints definiert werden wie im Deployment-Deskriptor (web.xml)
- Alle Ressourcen unter dieser URL sind geschützt z.B. mit «/SecureDecServlet/*»
- Details der Servlet-Spezifikation können auch unter http://download.oracle.com/otndocs/jcp/servlet-3_1-fr-eval-spec/index.html entnommen werden

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

```
package javax.servlet.annotation;
@Documented
@Retention(value=RUNTIME)

public @interface HttpConstraint {
    ServletSecurity.EmptyRoleSemantic value();
    java.lang.String[] rolesAllowed();
    ServletSecurity.TransportGuarantee transportGuarantee();
}
```

Annotation	Attribute	Beschreibung
@HttpConstraint	EmptyRoleSemantic	Definiert das Verhalten, wenn keine Rolle angegeben ist <ul style="list-style-type: none">Default = PERMITWerte EmptyRoleSemantic.PERMIT (alle Zugang) EmptyRoleSemantic.DENY (Keiner Zugang)Optional
	rolesAllowed	Komaseparierte Liste der erlaubten Rollen (Optional)
	transportGuarantee	Definiert ob eine Transportverschlüsselung gefordert ist oder nicht <ul style="list-style-type: none">Default = NONE (wenn Wert nicht angegeben wird)Werte NONE CONFIDENTIALOptional

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

```
package javax.servlet.annotation;
@Documented
@Retention(value=RUNTIME)

public @interface HttpMethodConstraint {
    ServletSecurity.EmptyRoleSemantic value();
    java.lang.String[] rolesAllowed();
    ServletSecurity.TransportGuarantee transportGuarantee();
}
```

Annotation	Attribute	Beschreibung
@HttpMethodConstraint	value	Name der HTTP-Methode (zwingend) [GET POST HEAD usw]
	EmptyRoleSemantic	Definiert das Verhalten, wenn keine Rolle angegeben ist <ul style="list-style-type: none">Default = PERMITWerte EmptyRoleSemantic.PERMIT (alle Zugang) EmptyRoleSemantic.DENY (Keiner Zugang)Optional
	rolesAllowed	Komaseparierte Liste der erlaubten Rollen (Optional)
	transportGuarantee	Definiert ob eine Transportverschlüsselung gefordert ist oder nicht <ul style="list-style-type: none">Default = NONE (wenn Wert nicht angegeben wird)Werte NONE CONFIDENTIALOptional

4.3

Sicherheit im Web

Programmatische Sicherheit

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
77



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

Java EE 7 Security: Reicht deklarative Sicherheit?

Wie kann man sich mit einem “nicht standardisierten” Verfahren authentisieren?



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
78

 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Programmatische Sicherheit kann verwendet werden, wenn die deklarative Sicherheit die Anforderungen nicht abdeckt.
- Programmatische und deklarative Sicherheit können kombiniert werden.

Java EE 7 Security: : Programmatische Sicherheit

Man bräuchte ein Login-Screen wie z.B:

The screenshot shows a web browser window with the URL http://localhost:8080/IEE_Kurs_Web_1-war/SecureServlet. The page content is:

Hello, please log in:

Please Enter Your User Name:

Please Enter Your Password:

Please Enter RSA Password and RSA Code:

Below the screenshot is a code snippet:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String j_username = request.getParameter("j_username");
    String j_password = request.getParameter("j_password");
    String j_rsa = request.getParameter("j_rsa");
    if (!login(j_username, j_password) && login(j_rsa)) {
        throw new SecurityException("User not logged in");
    }
}
```

On the right side of the slide, there is a watermark logo of the Swiss Confederation and text indicating the source of the material.

- In diesem Fall sind die Standard–Authentisierungsmechanismen nicht mehr ausreichend.
- Mehrere Möglichkeiten für eine Lösung
 - JAAS – Login Modul schreiben
 - Programmatische Lösung (Programmatische Sicherheit)
 - Weitere ...

Java EE 7 Security: Programmatische Sicherheit

Security-Methoden der Klasse HttpServletRequest	Beschreibung
void login(String username, String password)	Methode zum Einloggen des Benutzers (innerhalb des konfigurierten Realm)
void logout()	Methode zum Ausloggen des Benutzers (SecurityContext wird verworfen)
boolean authenticate(HttpServletResponse response)	Forciert ein Login, gemäss konfigurierten "login-config" Element (web.xml), durch den Webserver
boolean isUserInRole(String role)	Prüft ob Benutzer Inhaber einer Rolle ist
String getRemoteUser()	Gibt Username zurück (wenn Benutzer authentisiert ist)
Principal getUserPrincipal()	Gibt das User-Prinzipal zurück (wenn Benutzer authentisiert ist)
String getAuthType()	Gibt die Authentisierungsart zurück (BASIC_AUTH, FORM_AUTH, CERT_AUTH, DIGEST_AUTH oder null)

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
80



Swiss Federal Office of Justice and Police
Confédération suisse
Confederatio Helvetica
Confederatio svitza

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik ServiceCenter (ISC-JPO)
Abteilung Technologie
Peter Andri

- Die obenstehende Tabelle zeigt, welche Methoden innerhalb des «HttpServletRequest» dem Entwickler zur Verfügung stehen (JEE7).
- Siehe auch http://download.oracle.com/otn-pub/jcp/servlet-3_1-fr-eval-spec/servlet-3_1-final.pdf?AuthParam=1422724742_fbee58386ce4bf7ebceab72e8bf5a12b Kapitel 13.3 (Programmatic Security)

Java EE 7 Security: Programmatische Sicherheit

Weitere interessante Methoden (HttpServletRequest)	Beschreibung
<code>String getHeader("Authorization")</code>	Gibt in Fall von BASIC Authorizationen den Base64 kodierten Authentisierungsstring zurück (BASIC user:password)
<code>String getScheme()</code>	Gibt das verwendete Anfrage-Protokoll zurück, z.B. HTTP, HTTPS oder FTP
<code>isSecure()</code>	Gibt zurück, ob über SSL kommuniziert wurde oder nicht
<code>boolean isRequestedSessionIdValid()</code>	Prüft ob die aktuelle Session noch gültig ist

Java EE 7 Security: Programmatische Sicherheit

Listing: Die Verwendung von HttpServletRequest.isUserInRole

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    if (request.isUserInRole("manager")) {
        response.sendRedirect("/mgr/index.jsp");
    } else {
        response.sendRedirect("/guests/index.jsp");
    }
}
```

Listing: Die Verwendung von HttpServletRequest.login

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String userName = request.getParameter("user");
    String password = request.getParameter("password");
    try {
        request.login(userName, password);
    } catch (ServletException ex) {
        //Handling Exception
        return;
    }
}
```

Java EE 7 Security Übung 1: Web Security

- Importieren des «JEE7-Security_Kurs_Uebung_1.zip» Projektes in NetBeans
- Deklarierender benötigten Rollen (Definition des Mapping) im web.xml
- Absichern aller Webressourcen «/protected/*» mit HTTP BASIC Authentication Verwenden des erstellten Realms «SecureRealm» (Erlaubt sind die Rollen «administrator» und «user», gemäss Konfiguration) mit der Einstellung «default principal-to-role mapping»
- Eine Verschlüsselung ist nicht nötig
- Abhören des Datenverkehrs beim Login auf eine Ressource unter «/protected/*»
- Dabei können Tools wie TCP Monitor, Wireshark, Firebug oder anderer Browser-Entwicklungstool verwendet werden.
- Decodieren des «Authorization» HTTP-Headers (Base64 Encoded)

Beispiel eines HTTP-Header mit den Autorisierungsdaten

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.8,en;q=0.6
Authorization: Basic ZG9zdG86c29EOTIzNDUz
Connection: keep-alive
Host: localhost:8080
Referer: http://localhost:8080/JEE7-Secu_Web_1/ex/
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:36.0) Gecko/20100101 Firefox/36.0
```

- Zusatzaufgabe – Absichern mit HTTP form based Authentication, dabei können die Formulare «/login/loginform.html» und «/login/loginerror.html» verwendet werden oder eigene Formulare definiert werden
- Ausgeben des User, der Rolle, die Authentisierungsart und die Art der Verschlüsselung

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences



Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Decodieren kann man den Header z.B. unter <https://www.base64decode.org/>
- Vorsicht! Immer nur Testdaten Online dekodieren – niemals produktive Daten!
- Schauen Sie sich das loginform-Formular mit den entsprechenden Action und den entsprechenden Feldernamen an!
- Die Lösung befindet sich im Projekt «JEE7-Security_Kurs_Uebung_1_Loesung.zip»
- ! Wireshark kann unter Windows bei localhost-Verbindungen nicht verwendet werden, da Windows kein Loopback Interface kennt

5

Sicherheit im Web-Tier Advanced

Java EE 7 Security: combining constraints

Was passiert, wenn sich constraints überschneiden – sogenannte combining constraints? Es gelten folgende Regeln:

HTTP Methoden

- Ist eine HTTP-Methode (für die selbe URL) einmal geschützt und einmal ungeschützt, bleibt die HTTP-Methode **geschützt**.

Transportsicherheit

- Ein fehlendes `<user-data-constraint>` Element ermöglicht immer einen Zugriff ohne SSL (egal ob noch eine andere constraint für die entsprechende URL vorhanden ist)
- Wird für eine URL und eine HTTP-Methode einmal NONE und einmal CONFIDENTIAL verwendet, wird **keine Transportsicherheit (SSL)** forciert.

Java EE 7 Security: combining constraints

Rollen

- Sind in einer `<security-constraint>` keine Rollen für zugelassen (leeres `<auth-constraint>`) Element, können nicht in einem weiteren `<security-constraint>` Rollen erlaubt werden. Es werden keine Zugriffe gewährt
- Rollen von unterschiedlichen `<security-constraint>` für die selbe URL werden kumuliert (implizite Rollen `>*` und `**` Rollen ebenfalls)
- Eine `<security-constraint>` ohne ein Element `<auth-constraint>` ermöglicht immer einen unauthentisierten Zugriff auf die entsprechende URL (egal ob noch eine `<security-constraint>` mit einer `<auth-constraint>` für die URL vorhanden ist)
- Eine `<security-constraint>` mit einem leeren Element `<auth-constraint>` ermöglicht keinen Zugriff (egal ob noch eine `<security-constraint>` mit einer `<auth-constraint>` und Rollen für die URL vorhanden sind)

Java EE 7 Security: combining constraints

Verhaltens-Tipps mit "combining constraints"

- Überschneidungen wenn immer möglich **vermeiden** – da unter Umständen ein nicht deterministisches Verhalten provoziert werden kann (je nach Implementation, siehe auch JEE 7 Tutorial von Oracle)
- Sich gegenseitig ausschliessende constraints vermeiden (z.B. einmal keine Rollen für die HTTP-Methode POST zulassen – einmal die Rolle Admin für die selbe URL für die HTTP-Methode POST zulassen). Dies kann zu nicht erwünschten Effekten führen.
- Wird SSL gefordert mit dem Element <user-data-constraint> (CONFIDENTIAL), ist darauf zu achten, dass bei Überschneidungen der security-constraint (URL-Pattern) immer CONFIDENTIAL gewählt wird, sonst wird die Ressource auch über ungeschützten Transport zugänglich.
- Vorsicht bei SSL – wird einmal auf SSL gewechselt darf nicht mehr auf non-SSL zurückgewechselt werden (Sicherheit !)

Java EE 7 Security Übung 2: combining constraints

- ▶ Analysieren sie die folgenden 4 <security-constraint>
- ▶ Füllen sie die folgende Tabelle aus (SecurityConstraint.xlsx)
- ▶ Markieren sie die Verbindungen, welche durch SSL geschützt sind
- ▶ Präsentieren und diskutieren Sie Ihre Lösungen

<security-constraint>								
<security-constraint>	<url-pattern>	GET	POST	PUT	HEAD	TRACE	DELETE	OPTIONS
1. constraint	/* /foo/* /bar/*							
2. constraint	/foo/*							
3. constraint	/foo/*							
4. constraint	/bar/*							

Zugriffe (Resultate aller constraints)								
	/*							
	/*							
	/foo/*							
	/bar/*							

Rollen	USER	ADMIN	OWNER
	Ohne	Mit	

Berner Fachhochschule Haute école spécialisée bernoise Bern University of Applied Sciences 90	 Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra	Eidgenössisches Justiz- und Polizeidepartement (EJPD) Informatik Service Center (ISC-EJPD) Abteilung Technik Peter Andri
--	---	---

- Die Vorlage befindet sich im Dokument SecurityConstraint.xlsx
- Die Lösung befindet sich im Dokument SecurityConstraintLoesung.xlsx
- Zeitbedarf 15'

Java EE 7 Security Übung 3: combining constraints

<security-constraint> Script 1

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Excluded methods</web-resource-name>
    <url-pattern>/*</url-pattern>
    <url-pattern>/foo/*</url-pattern>
    <url-pattern>/bar/*</url-pattern>
    <http-method-omission>GET</http-method-omission>
    <http-method-omission>POST</http-method-omission>
  </web-resource-collection>
  <auth-constraint/>
</security-constraint>
```

Java EE 7 Security Übung 3: combining constraints

<security-constraint> Script 2

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>User Ressource</web-resource-name>
    <url-pattern>/foo/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>PUT</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>USER</role-name>
  </auth-constraint>
</security-constraint>
```

Java EE 7 Security Übung 2: combining constraints

```
<security-constraint> Script 3

<security-constraint>
  <web-resource-collection>
    <web-resource-name>User Ressource</web-resource-name>
    <url-pattern>/foo/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>ADMIN</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transportguarantee>
  </user-data-constraint>
</security-constraint>
```

Java EE 7 Security Übung 3: combining constraints

<security-constraint> Script 4

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Admin Ressource</web-resource-name>
    <url-pattern>/bar/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>ADMIN</role-name>
    <role-name>OWNER</role-name>
  </auth-constraint>
</security-constraint>
```

Java EE 7 Security Übung 3: LoginModul I

Custom-LoginModul in Glassfish 4.0

- Ziel ist es, dass sich der User mit einem SecureID-Token authentisieren kann
- Dazu soll keine programmatische Lösung (im Sinne innerhalb der Applikation) erarbeitet werden, sondern ein Custom JAAS – Loginmodul für den Glassfish 4.x
- Um mit einer SecureID einloggen zu können braucht es in der Regel einen Radius Server
- Dieser Radius Server wird «gemockt» mit Hilfe der Klassen
 - ch.bfh.loginmodule.RadiusAuthentication
 - ch.bfh.loginmodule.RadiusMock (implements RadiusAuthentication)
- Die Methode
 - public boolean authenticate(String username, String password)
 - kann verwendet werden um den User zu authentisieren
- Die Mock-Klasse enthält statische User (user1)
- Die Mock-Klasse erwartet Passwörter gemäss folgendem Pattern
 - HHMM12345678, wobei HHMM die aktuelle Zeit in Stunden (24h) und Minuten darstellt.

- Der Glassfish Server nimmt schon einiges an Arbeit ab – mit der Implementierung der Klassen com.sun.appserv.security.AppservPasswordLoginModule und com.sun.appserv.security.AppservRealm können LoginModule einfach erzeugt werden
- Die Lösung befindet sich unter GlassFishLoginModuleLoesung.zip

Weitere Info siehe auch :

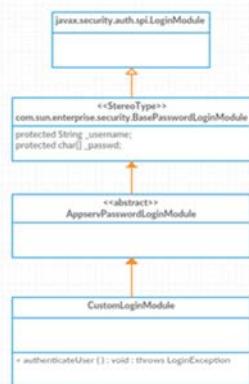
- <http://byorns.blogspot.ch/2015/01/how-to-setup-custom-jaas-login-module.html>
- https://blogs.oracle.com/nithya/entry/groups_in_custom_realms
- Zeitbedarf 15'

Java EE 7 Security Übung 3: LoginModul I

Anleitung für ein Custom-LoginModul in Glassfish 4.0

Login-Formular (FORM Based authentication)

- Absichern der Applikation mit HTTP FORM Authentication (siehe Übung 2) - ev. ergänzen des Text für das Passwort (HHMM+PW).



JAAS LoginModule und Realm

- Erzeugen des CustomLoginModules (2 Dateien sind nötig)
 - 1) Eine CustomLoginModule - Klasse welche die Klasse `com.sun.appserv.security.AppservPasswordLoginModule` implementiert
 - 2) Eine CustomRealm Klasse welche die Klasse `com.sun.appserv.security.AppservRealm` implementiert
- Um kompilieren zu können, müssen im Netbeans-Projekt folgende Libraries eingebunden werden
 - `glassfish-ee-api.jar` (unter `glassfish-4.1\glassfish\modules`)
 - `security.jar` (unter `glassfish-4.1\glassfish\modules`)

- Der Glassfish Server nimmt schon einiges an Arbeit ab – mit der Implementierung der Klassen `com.sun.appserv.security.AppservPasswordLoginModule` und `com.sun.appserv.security.AppservRealm` können LoginModule einfach erzeugt werden
- Die Lösung befindet sich unter `GlassFishLoginModuleLoesung.zip`

Weitere Info siehe auch :

- <http://byorns.blogspot.ch/2015/01/how-to-setup-custom-jaas-login-module.html>
- https://blogs.oracle.com/nithya/entry/groups_in_custom_realms
- Zeitbedarf 15'

Java EE 7 Security Übung 3: LoginModul II

Klasse CustomLoginModule

- CustomLoginModule welches AppservPasswordLoginModule implementiert
- Überschreiben der Methode authenticateUser in der Klasse CustomLoginModule (die Attribute _username und _passwd können verwendet werden)
- Am Ende (wenn erfolgreich authentisiert) der Methode authenticateUser muss dem User die entsprechende Gruppe (Rolle) zugewiesen werden

```
String[] groups = {"user"};
commitUserAuthentication(groups);
```

Klasse CustomRealm

- CustomRealm welches AppservRealm implementiert
- Überschreiben der Methode init in der Klasse CustomRealm und setzen des JAAS_CONTEXT («jaas-context»)

```
String propJaasContext=properties.getProperty(JAAS_CONTEXT);
if (propJaasContext!=null) {
    setProperty(JAAS_CONTEXT, propJaasContext);
}
```

Beide Klassen-Body sind bereits im GlassFishLoginModule.zip vorbereitet

<http://byorns.blogspot.ch/2015/01/how-to-setup-custom-jaas-login-module.html>

https://blogs.oracle.com/nithya/entry/groups_in_custom_realms

Java EE 7 Security Übung 3: LoginModul III

Deployen des Loginmodules (CustomLoginModul)

- ▶ Kopieren des jar-Files (CustomLoginModul) nach
 \glassfish\domains\domainXYZ\lib
- ▶ Editieren der Datei login.conf (
 \glassfishinstallation\glassfish\domains\domainXYZ\config)
- ▶ Registrieren des Realms

```
customloginmoduleRealm {  
    customloginmodule.CustomLoginModule required;  
};
```

Konfigurieren des Glassfish (in der Konsole)

- ▶ Erzeugen eines neuen Realm im Glassfish mit dem Namen
 «customRealm»
- ▶ Erfassen eines Property mit dem Namen “jaas-context” und dem Wert
 customLoginModuleRealm, welches so konfiguriert wurde in der Datei
 login.conf
- ▶ Testen ☺

6

Sicherheit im EJB-Tier

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
103



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederatio svitza

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

6.1

Sicherheit im EJB Tier

Grundlagen

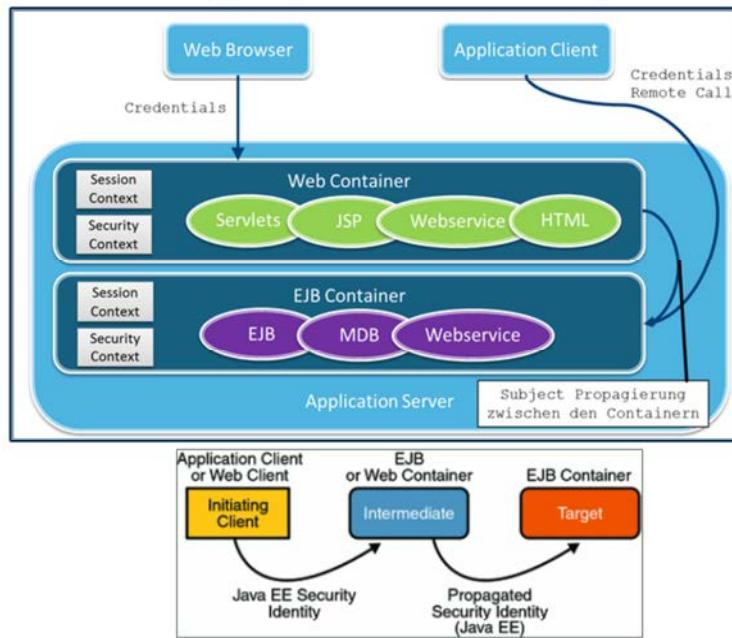
Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
104



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

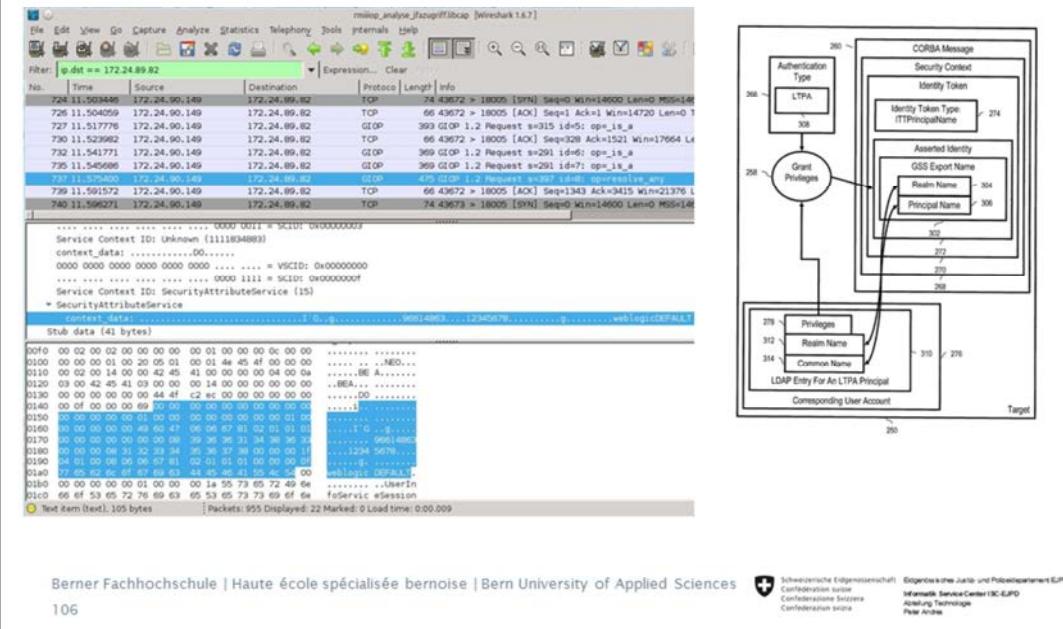
Java EE 7 Security – EJB-Tier: Grundlagen



- Zwischen Web- und EJB-Container wird keine Authentisierung mehr vorgenommen (Container vertrauen sich innerhalb des Applikationsserver [Glassfish]).
- Bei Aufrufen zwischen unterschiedlichen Applikationsservern werden Subject und Credentials im GIOP - Protokoll (IIOP) mitgegeben (siehe Wireshark Auszug).
- Wie im Web-Container können wir im EJB-Container die Security Deklarativ per Annotation, Deploymentdeskriptor und / oder programmatisch definieren.
- In einem EJB Modul können wir Sicherheitsaspekte definieren für Entity Beans und für Session Beans (inkl. annotierte Webservices), nicht aber für MDBs.
- RMI / IIOP ist in JEE 8 “pruned”

Java EE 7 Security – EJB-Tier: Grundlagen

Wireshark – Mitschnitt (Oracle-Weblogic-Server 12.2)



- Beispiel eines Whire-Shark-Mitschnitts von einem Remote-Call (Servlet <-> Remote EJB)
- Details zu GIOP (General Inter-ORB Protocol), CORBA (Common Object Request Broker Architecture), IOP (Internet Inter-ORB Protocol) und CSIV2 (Common Secure Interoperability) siehe auch
 - <http://www.omg.org/spec/>
 - <http://www.omg.org/spec/CORBA/3.3/Interoperability/PDF/>
 - <http://de.wikipedia.org/wiki/GIOP>, <http://de.wikipedia.org/wiki/IOP>

6.1

Sicherheit im EJB Tier

Deklarative Sicherheit

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
107

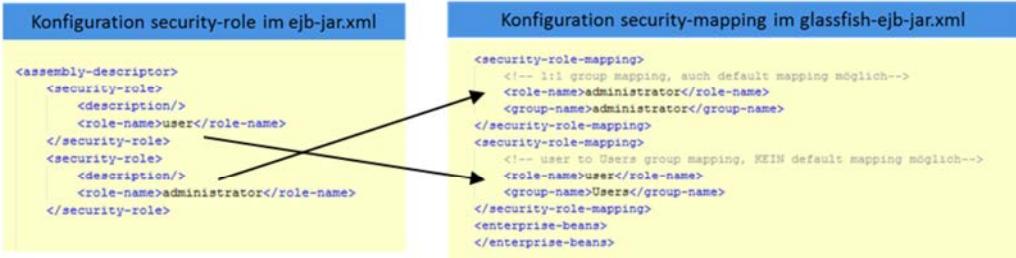


Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

**Alle verwendeten Rollen sind zu deklarieren im `ejb-jar.xml`
Deskriptor**



- Wie beim «glassfish-web.xml» kann auch ein User-Mapping im «glassfish-ejb-jar.xml» gemacht werden (wenn nötig).
- Die Namen der Rollen im «ejb-jar.xml» müssen identisch sein mit den Namen in der Mapping-Datei (glassfish-ejb-jar.xml)

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Absichern (method-permission) von EJBs via Deployment-Deskriptor (ejb-jar.xml)

Listing: method-permission im ejb-jar.xml

```
<method-permission>
  <description>Der Administrator darf alle RemoteServiceBean-Methoden aufrufen </description>
  <role-name>Administrator</role-name>
  <method>
    <ejb-name>RemoteServiceBean</ejb-name>
    <method-name>*</method-name>
  </method>
</method-permission>
<method-permission>
  <description>Die Rolle User darf alle nur die readService-Methoden aufrufen </description>
  <role-name>Administrator</role-name>
  <role-name>User</role-name>
  <method>
    <ejb-name>RemoteServiceBean</ejb-name>
    <method-name>readService</method-name>
    <method-params>
      <method-param>java.lang.String</method-param>
    </method>
  </method>
</method-permission>
```

Rollen (1-n), welche für diese Ressource zugelassen sind

Geschützte Ressourcen (EJB)

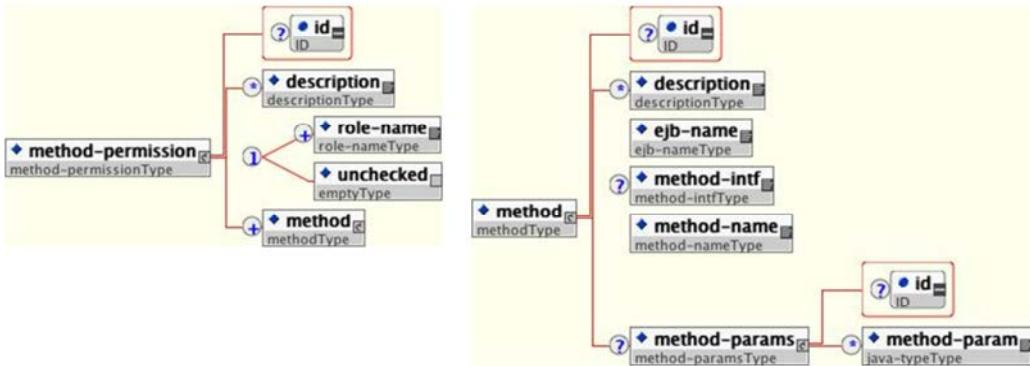
Geschützte Methode der Ressource (EJB) [* = alle]

Parameter der geschützten Methode

- Deklarative Sicherheit im Deployment-Deskriptor (ejb-jar.xml) wird eher selten verwendet.
- Pro EJB (ausser MDB) können eine oder mehrere «method-permission» erzeugt werden.
- Im Element «ejb-name» wird das zu schützende EJB (ausser MDB) angegeben.
- Im Element «method-name» wird die zu schützende Methode (ev. mit Parameter) angeben.
- Mit dem Element «unchecked» können Methoden des EJB ausgenommen werden.
- Deklarationen im Deployment-Deskriptor überschreiben die Annotationen in den Klassen.
- Weitere Konfigurationen sind möglich (siehe Schema http://xmlns.jcp.org/xml/ns/javaee/ejb-jar_3_2.xsd).

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Schemas



Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Absichern (method-permission) von EJB via Annotationen

Listing: Annotation in der Klasse SecureSessionBean.java

```
@Stateless
@LocalBean
//nur nötig, wenn innerhalb des EJB die Methode "isCallerInRole" verwendet wird
@DeclareRoles({"user", "administrator"})
public class SecureSessionBean implements ISecureSessionBean {

    @Override
    @RolesAllowed({"user", "administrator"}) //User und Administrator erlaubt
    public String secureUserEcho(String echo) {
        return echo;
    }

    @Override
    @RolesAllowed("administrator") //nur der Administrator
    public String secureAdminEcho(String echo) {
        return echo;
    }

    @Override
    @PermitAll //erlaube allen
    public String secureEcho(String echo) {
        return echo;
    }

    @Override
    @DenyAll //erlaube niemand
    public String internSecureEcho(String echo) {
        return echo;
    }
}
```

Definition der verwendeten Rollen

Rollen (1-n), welche für diese Ressource zugelassen sind

Alle Rollen sind für diese Methode zugelassen

Keine Rollen sind für diese Methode zugelassen

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
111



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technik
Peter Andri

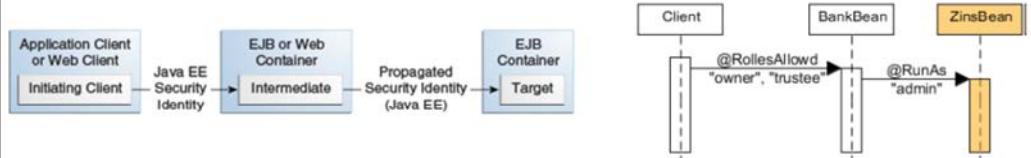
- Deklarative Sicherheit mit Annotationen werden häufig verwendet
- `@DeclareRoles` beschreibt alle verwendeten Rollen. Optional, wenn die Methode `isCallerInRole()` **nicht** verwendet wird.
- `@RolesAllowed` kann 1-n Rollen enthalten. Diese müssen, wie im Web-Tier, auf den User/ Gruppe gemapped werden können.
- Deklaration auf der Methode überschreibt die Deklaration auf der Klasse
- Deklarationen auf der Klasse gelten für alle Methoden
- Hinweis: Annotationen werden nicht vererbt – die Annotationen gelten immer für die Klasse / Methoden, in welcher diese deklariert wurden.
- Vorsicht – wenn eine Methode innerhalb der Klasse aufgerufen wird, werden die Annotationen nicht durchlaufen. Unten stehendes Beispiel kann von «aussen» nicht aufgerufen werden, wohl aber innerhalb der Klasse:

```
@DenyAll
private void test() {
    //do something
}
```

- Nicht alle Security-Einstellungen können per Annotationen erstellt werden – so kann z.B. im EJB Container der Authentisierungs-Mechanismus nicht per Annotation definiert werden. Dies muss über den (proprietären) Deployment-Deskriptor passieren. Der Standard Authentisierungs-Mechanismus im Glassfish-Applikationsserver (EJB Container) ist BASIC (username / password).
- Ist eine «method-permission» definiert, verlangt der Glassfish eine Authentisierung (Username / Passwort)
- Detail zu den Annotationen siehe auch <https://jcp.org/en/jsr/detail?id=250>

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Verwenden der „run-as“ Annotationen zur «Rollen-Propagierung»



```
@Stateless
@LocalBean
@DeclareRoles({"user", "administrator"})
@RunAs("Manager")
public class SecureSessionBean implements ISecureSessionBean {

    @Resource
    private SessionContext sctx;

    @EJB
    private RemoteServiceBean rsb;

    @Override
    public String secureEcho(String echo) {
        //Outgoing Call -> propagiert wird die Rolle "Manager"
        rsb.service(echo);
    }
}
```

Die «run-as» Annotation wird verwendet, um eine spezifische Rolle bei einem ausgehenden Call zu definieren. Dies ermöglicht es, zwischen Intermediate EJB Container und Target EJB Container unterschiedliche Rollen zu verwenden. Könnte z.B. auch in einem Test (wo der User nicht alle Rollen besitzt) angewendet werden.

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Security-Annotation für Web- und EJB-Container

Annotation	Beschreibung
@ServletSecurity	Das @ServletSecurity kann @HttpMethodConstraint und @HttpConstraint Elemente enthalten für die Absicherung von Servlets.
@DeclareRoles	DeclareRoles definiert analog dem Element <security-role> die verwendeten Rollen.
@RunAs	Definiert die Rolle für ausgehende Aufrufe
@PermitAll	Erlaubt allen Rollen auf die Methode(n) zuzugreifen
@DenyAll	Erlaubt keiner Rolle auf die Methode zuzugreifen
@RolesAllowed	Definiert welche Rollen Zugriff auf die Methode(n) haben

Java EE 7 Security: Deklarative Sicherheit (Deskriptor)

Security-Annotation für Web- und EJB-Container

Annotation	Targets Level	Target Kind
@ServletSecurity	Class	Servlet
@DeclareRoles	Class	EJB, Servlet
@RunAs	Class	EJB, Servlet
@PermitAll	Class, Method	EJB
@DenyAll	Method	EJB
@RolesAllowed	Class, Method	EJB

6.3

Sicherheit im EJB Tier

Programmatische Sicherheit

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
115



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

Java EE 7 Security: Programmatische Sicherheit

Listing: Die Verwendung von javax.ejb.EJBContext

```
@Resource
private SessionContext sctx;

@Override
public String secureEcho(String echo) {

    String name = sctx.getCallerPrincipal().getName();
    if (sctx.isUserInRole("user")) {
        //Business für die Rolle "user"
        requestUserData(name);
    } else if (sctx.isUserInRole("administrator")) {
        //Business für die Rolle "administrator"
        requestAdminData(name);
    } else {
        //Business für alle anderen Rolle .
        requestAnonymousData(name);
    }
    return echo;
}
```

Auslesen des aktuellen Users aus dem Prinzipal

Prüfen, ob ein User in einer bestimmten Rolle ist

Ausliefern von Daten anhand der Rolle, welche der Benutzer besitzt

- Wie beim Web-Teil kann programmatische Sicherheit verwendet werden, wenn die deklarative Sicherheit (Deployment-Deskriptor resp. Annotationen) nicht ausreichend sind.
- Der «javax.ejb.SessionContext» resp. der «javax.ejb.EJBContext» kann verwendet werden, um einerseits den aktuellen Benutzer zu bekommen (`getCallerPrincipal().getName()`) und andererseits zu testen, ob der aktuelle Benutzer eine spezifische Rolle besitzt (`isUserInRole()`).
- Wird die Methode `isUserInRole()` verwendet, müssen die Rollen in der Klasse per Annotation definiert werden (`@DeclareRoles`)
- Die Credentials können nicht über ein Standard-API (der JEE Plattform) ausgelesen werden. Gewisse Applikations-Server bieten proprietäre Erweiterungen

Java EE 7 Security: Programmatische Sicherheit

Security-Methoden der Klasse EJBContext, resp. SessionContext	Beschreibung
public Principal getCallerPrincipal()	Gibt das aktuelle Prinzipal des Aufrufers zurück
String java.security.Principal.getName();	Gibt den Namen des Prinzipals zurück
public boolean isCallerInRole (String roleName)	Prüft zur Laufzeit, ob der aktuelle User die Rolle «roleName» besitzt

Java EE 7 Security: Vergleich

Web Tier	EJB Tier		deklarative Sicherheit	programmatische Sicherheit
Web Resources	Bean Methods	Zugriffskontrolle	Deployment Descriptor	Programm
web.xml	ejb-jar.xml	Deklaration		
Web Container	EJB Container	Umsetzung	Container	Programm
Servlet / JSP	EJB Bean	Codierung	"all or nothing"	Logikbasiert

Java EE 7 Security Übung 4: EJB Security

- Importieren des «JEE7-Security_Kurs_Uebung_4.zip» Projektes in NetBeans
- Absichern des SecureEJB, dabei soll folgendes gelten:
 - echo() dürfen alle aufrufen
 - readInternalData() dürfen alle Benutzer der Gruppe «user» und «administrator» aufrufen
 - readConfidentialData() dürfen nur Benutzer der Gruppe «administrator» aufrufen
- Zeigen Sie 2 Lösungsmöglichkeiten (deklarativ, programmatisch)
- Loggen Sie den Zugriff. Im Log soll enthalten sein
 - User (Username)
 - Methode mit Klassenname (ohne Parameter)
- Die Lösung befindet sich «JEE7_Security_Kurs_Uebung_3_Loesung.zip»

- Die Lösung finden sie unter JEE7-Security_Kurs_Uebung_4._Loesungzip

7

Sicherheit Webservices (im EJB Container)

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
122



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

- ▶ **EJB-Container (Webservice als EJB implementiert)**
- ▶ Bei annotierten Webservices im EJB-Tier (stateless Sessionbeans) können Annotationen wie `@RolesAllowed`, `@PermitAll` usw. verwendet werden.
- ▶ Jedoch ist die Definition des Loginverfahrens (Verfahren, Realm usw) proprietär gelöst.

Listing: Absichern eines Webservice (JBoss proprietär per Annotation)

```
@Stateless
@WebService(serviceName = "SecureWebService")
@WebContext(contextRoot="/SecureService",
            urlPattern="/Service",
            authMethod="BASIC",
            secureWSDLAccess = false)

@SecurityDomain(value = "jboss-sec-domain")
public class SecureWebService {
```

Listing: Absichern eines Webservice (Glassfish proprietär, glassfish-ejb-jar.xml)

```
<enterprise-beans>
  <ejb>
    <ejb-name>SecureWebService</ejb-name>
    <web-service-endpoint>
      <port-component-name>Service</port-component-name>
      <login-config>
        <auth-method>BASIC</auth-method>
        <realm>SecureRealm</realm>
      </login-config>
    </web-service-endpoint>
  </ejb>
</enterprise-beans>
```

Web-Container (Webservice als Servlet)

Webservices im Web-Tier können wie «normale» Webressourcen im Deployment-Deskriptor (web.xml) abgesichert werden.

- Webservices im Web-Tier können gemäss Kapitel 4 «ganz normal» abgesichert werden – hier sprechen wir von annotierten Webservices im EJB Tier-
- Keine standardisierten Annotationen für die Sicherheit bei annotierten Webservices im EJB-Container.
- Es gibt proprietäre Möglichkeiten, Annotationen oder Deployment-Deskriptoren zu verwenden (siehe Listing JBOSS und Glassfish).
- Nicht zu verwechseln mit WS-Security (XML Signature, XML Encryption), siehe auch «Messagebasierte Sicherheit».
- Messagebasierte Sicherheit siehe auch https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- Webservices im Web-Container können standardisiert via Deklarationen im web.xml mit einem entsprechenden «security constraint» erfolgen.

8

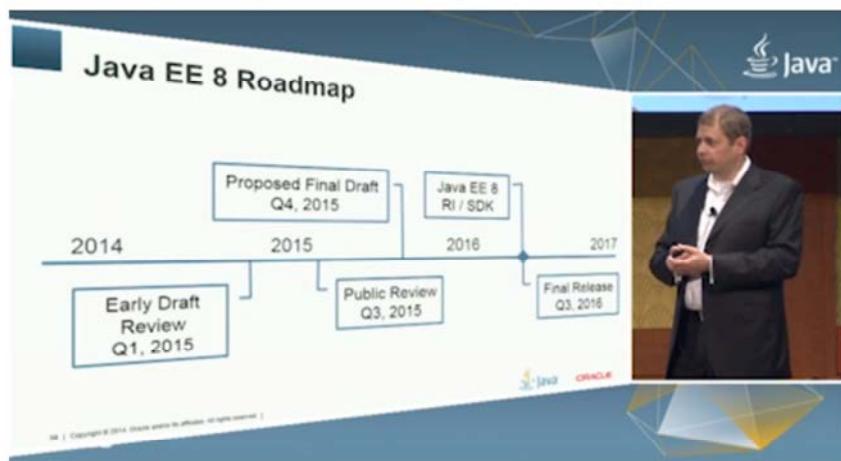
Ausblick JEE8 (Sicherheit)

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
124



Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

Ausblick JEE8 (Sicherheit)



Quelle Oracle

... we are therefore publicly announcing that we are now changing our target time frame for the completion of this **work to the first half of 2017**.

Ausblick JEE8 (Sicherheit)

Neuer JSR (Java Specification Request) für Java Security: JSR 375 - Java EE Security 1.0

- ▶ Das Java EE Security 1.0 (JSR 375) API stellt Einbindung eines vereinfachten Sicherheits-API dar, das Java-EE-Anwendungen die Verwaltung ihrer eigenen Sicherheitsparameter in einzigartiger und doch portabler Art und Weise ermöglicht.
- ▶ Dieser JSR wird ebenfalls für Cloud-basiert Java-EE-Anwendungsbereitstellungen nützlich sein, die auf der Suche nach einer Standardmethode für die Definition von Sicherheitsaspekten sind.
- ▶ Das JSR 375 Proposal (<https://www.jcp.org/en/jsr/detail?id=375>) gibt eine grobe Übersicht, über die Inhalte des neuen API.

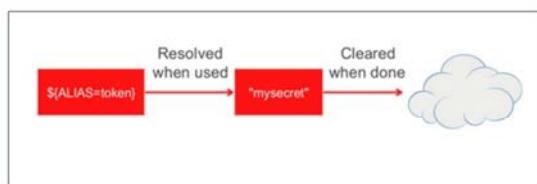
- Siehe auch:
 - https://blogs.oracle.com/theaquarium/entry/jsr_375_java_ee_security
 - https://blogs.oracle.com/theaquarium/entry/javaone_replay_java_ee_8
 - https://blogs.oracle.com/theaquarium/entry/jsr_375_java_ee_security

Ausblick JEE8 (Sicherheit) JSR 375

Password aliasing

- Standardisierter Syntax für das “password aliasing”
 - Ermöglicht das Weglassen von Klartext-Passwörter im Code

```
@DataSourceDefinition(  
    name="java:app/jdbc/test",  
    user="root",  
    password="${ALIAS=password}",...)  
  
    • For deployment descriptors  
  
<data-source>  
<name>java:app/env/testDS</name>  
<user>APP</user>  
<password>${ALIAS=password}</password>
```



Ausblick JEE8 (Sicherheit) JSR 375

User Management

- ▶ Erlaubt Applikationen ihre eigenen User und Gruppen zu administrieren
 - ▶ Ohne die Applikations-Server-Konfiguration zu verändern
- ▶ User werden in einem applikations-spezifischen Repository (z.B. LDAP) gespeichert
- ▶ Der User-Service [UserService] erlaubt
 - ▶ Erzeugen/Löschen von User und Gruppen, ändern von User / Gruppen, auslesen von Namen usw
- ▶ UserInfo enthält Infos wie Name, Credentials, Rolle(n), weitere Attribute



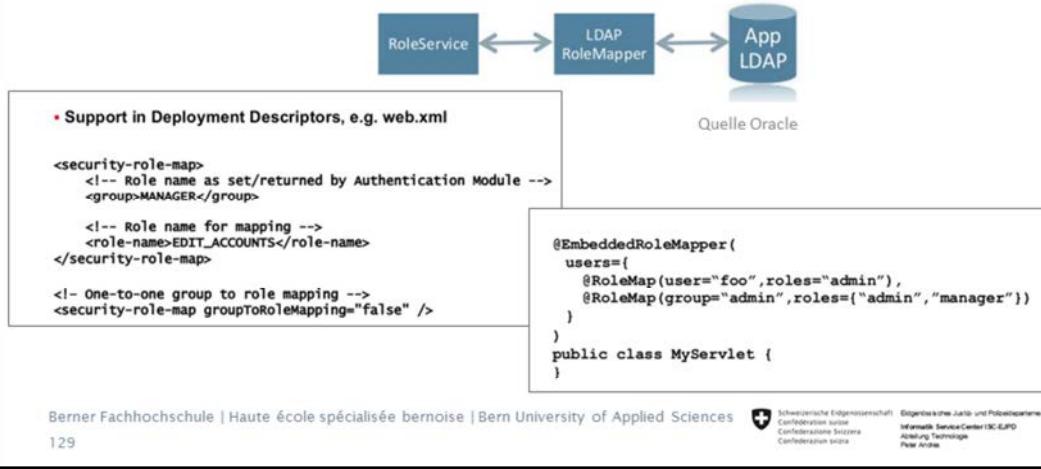
Quelle Oracle

```
@LdapUserSourceDefinition(  
    name="java:app/ldapUserSource",  
    ldapUrl="ldap://someURL",  
    ldapUser="ElDap",  
    ldapPassword="${ALIAS=LdapPW}",  
    ...  
)  
public class MyAuthenticator {  
    @Resource(lookup="java:app/ldapUserSource")  
    private UserService userService;  
    private boolean isAccountEnabled(String username) {  
        return userService.loadUserByUsername(username).isEnabled();  
    }  
    ...  
}
```

Ausblick JEE8 (Sicherheit) JSR 375

Standardisiertes User-Role-Mapping

- ▶ Role-Mappings können in einem applikations-spezifischen Repository gespeichert werden
- ▶ Applikationen können Rollen und Gruppen zu User mappen anhand anwendungsspezifischen Use-Cases
- ▶ Alles ohne die Hilfe des Applikations-Servers



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
129



Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

Ausblick JEE8 (Sicherheit) JSR 375

Authorization Interceptors

- Die Authentisierung kann neu nicht nur anhand von Rollen definiert werden, sondern es kann eine rule-based security annotiert werden

https://blogs.oracle.com/theaquarium/entry/javaone_replay_java_ee_8

- EL Authorization rules centrally managed in a repository

```
@LdapAuthorizationRules (name="java:app/accountAuthRules",  
    ldapUri="ldap://blah",  
    ldapUser="ElDap",  
    ldapPassword="mysecret")  
public class MyBean {  
    @EvaluateSecured(  
        ruleSourceName="java:app/accountAuthRules", rule="transferFunds")  
    void transferFunds() {  
        ...  
    }  
}  
  
@IsAuthorized("hasRoles('Manager') && schedule.officeHours")  
public void transferFunds() {  
    ...  
}  
  
@IsAuthorized(  
    "hasRoles('Manager') && hasAttribute('directReports', employeeId)")  
public double getSalary(long employeeId) {  
    ...  
}  
  
@IsAuthorized(ruleSourceName="java:app/payrollAuthRules")  
public void displayReport() {  
    ...  
}
```

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
130



Schweizerische Eidgenossenschaft
Confédération Suisse
Confederatio Helvetica

Eidgenössisches Justiz- und Polizeidepartement EJPD
Informatik Service Center ISC-EJPD
Abteilung Technologie
Peter Andri

7

Transportsicherheit Secure Sockets Layer

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
131



Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

7.1

Transportsicherheit

Grundlagen – und grobe Übersicht

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
132

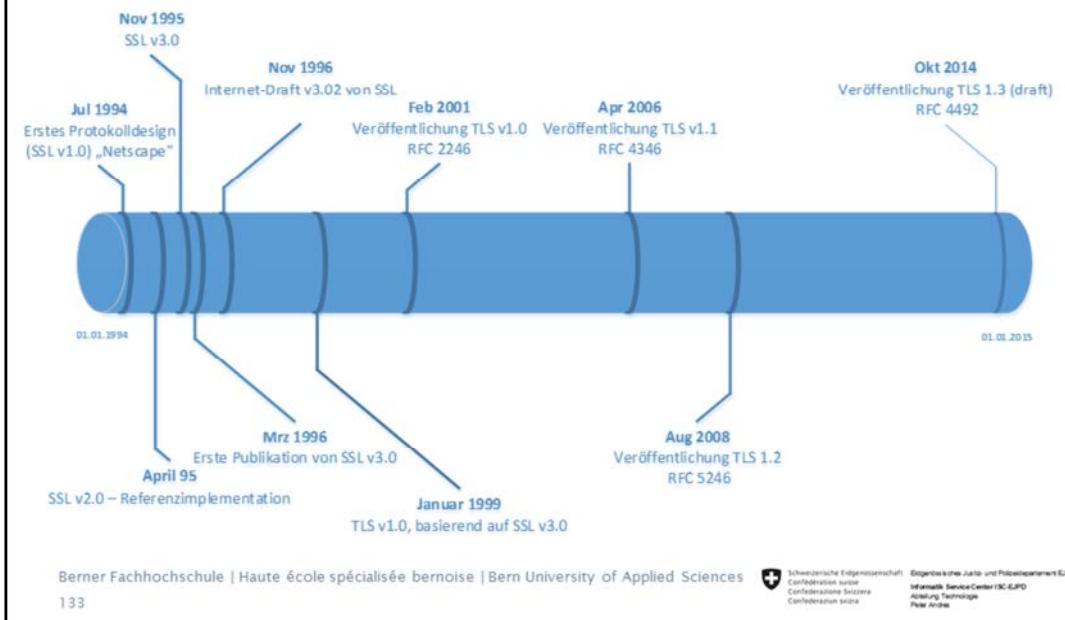


Schweizerische Eidgenossenschaft
Confédération suisse
Confederatio Helvetica
Confederatio svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

Java EE 7 Security – Transportsicherheit (SSL / TLS)

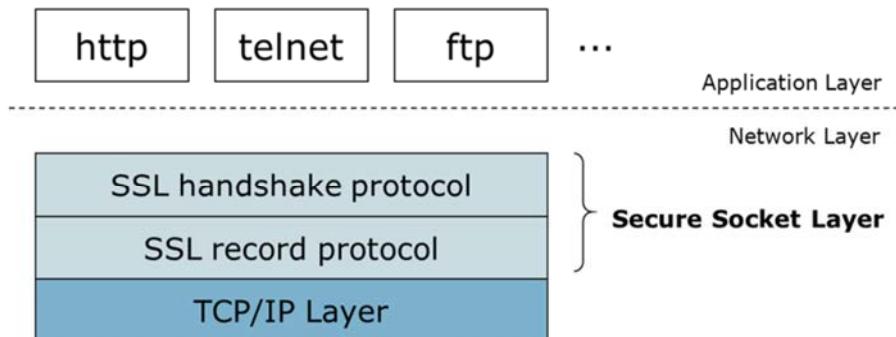
Secure Sockets Layer (SSL) resp. Transport Security Layer (TLS) Geschichte



- SSL – durch Netscape ins Leben gerufen – ist heute der Standard um Datenkommunikation zu verschlüsseln.
- Heute sollten alle aktuellen Webserver TLS 1.2 unterstützen – SSLV3 und tiefere Versionen sollten nicht mehr verwendet werden.

Java EE 7 Security – Transportsicherheit (SSL / TLS)

OSI Modell Layer 3 / 4 (Transport / Netzwerk)

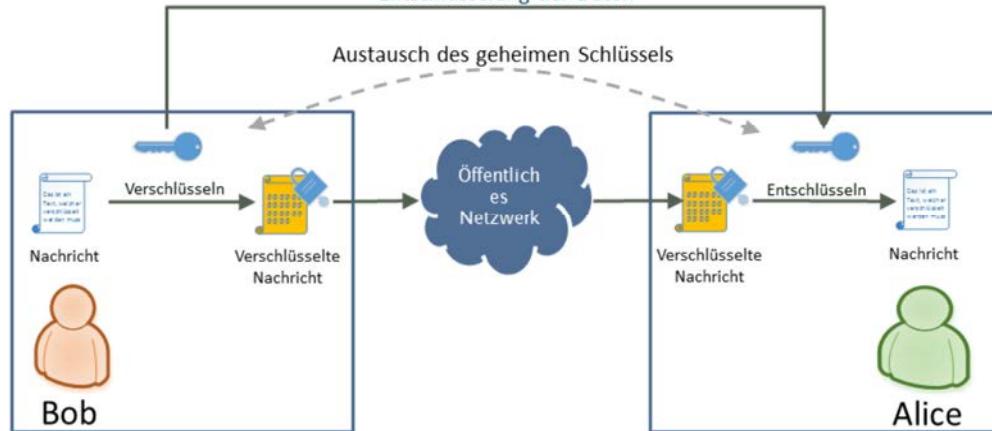


- SSL befindet sich auf dem Netzwerk-Layer, im OSI Modell auf Layer 3 / 4
- SSL ist also für die Applikationen (z.B. Webapplikationen, welche HTTP Protokolle verwenden) transparent
- Der Overhead ist heute für SSL sehr klein (im Verhältnis zur Gesamttransaktion). Der grösste Overhead besteht im SSL-Handshake.

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Symmetrisches Verschlüsselungs-Verfahren

Ein Schlüssel für die Ver- und Entschlüsselung der Daten



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
135

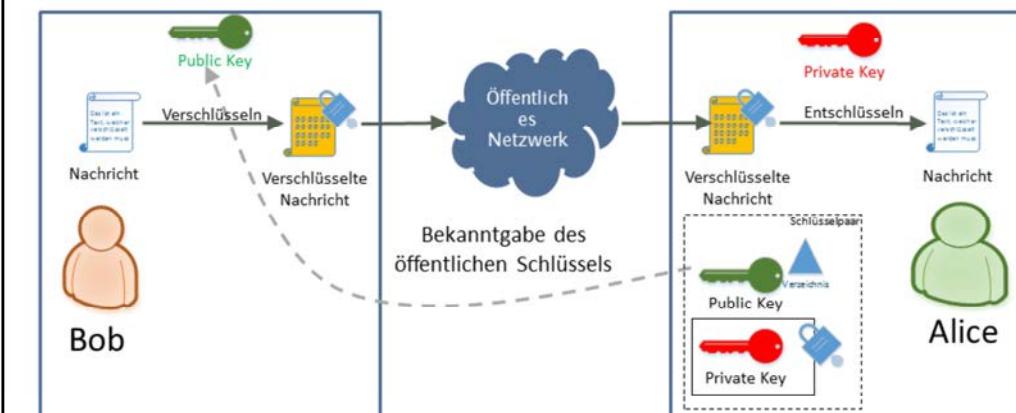
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technik
Peter Andri

- Bei der symmetrischen Verschlüsselung kennen und benutzen Sender und Empfänger für die Ver- und Entschlüsselung einer Nachricht den gleichen geheimen Schlüssel.
- Dieses Verfahren ist sehr schnell.
- Der grosse Nachteil ist, dass der Schlüsselaustausch über einen öffentlichen Kanal schwierig ist.
- Typische Algorithmen sind DES, 3DES, AES, blowfish, twofish ...

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Asymmetrisches Verschlüsselungs-Verfahren



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
136

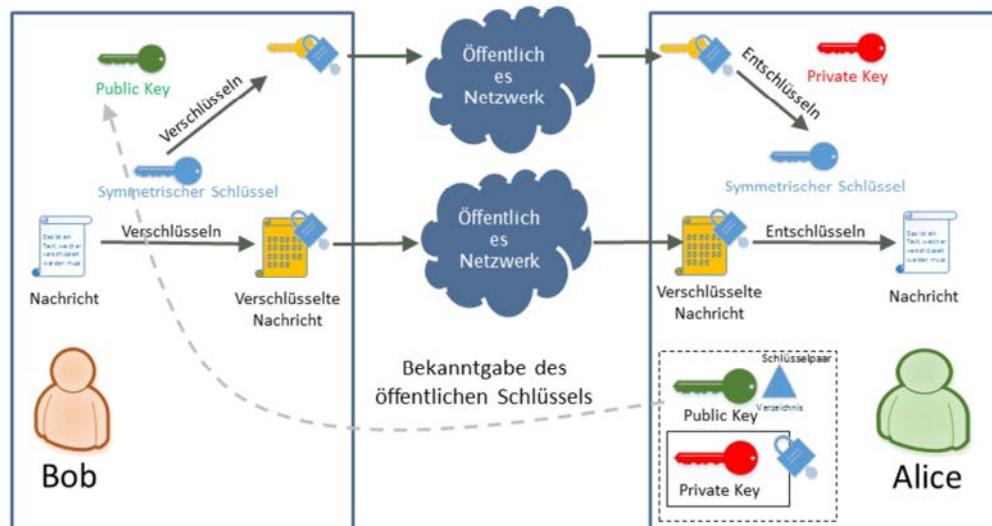
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Das Konzept der asymmetrischen Verschlüsselung wurde 1976 von Whitfield Diffie und Martin Hellman vorgeschlagen, um das Problem zu lösen, dass bei symmetrischer Verschlüsselung der geheime Schlüssel zwischen den Kommunikationspartnern ausgetauscht werden muss.
- Jeder Beteiligte hat zwei Schlüssel, einen öffentlichen (Public Key) und einen privaten (Privat Key). Vom einen Schlüssel lässt sich niemals auf den anderen Schlüssel schliessen.
- Private und Public Key sind komplementär zueinander.
- Der private Schlüssel wird niemals preisgegeben.
- Das Verfahren kann verwendet werden, um Daten zu signieren (privater Schlüssel, Verifikation öffentlicher Schlüssel) und Daten zu verschlüsseln (öffentlicher Schlüssel, entschlüsseln privater Schlüssel)
- Dieses Verfahren ist langsam (Faktor 1000 – 10'000 x langsamer als symmetrische Verschlüsselung)
- Typische Algorithmen sind DH (Diffie-Hellman), RSA (Rivest, Shamir und Adleman), DAS (Digital Signature Algorithm) usw.

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Hybrides Verschlüsselungs-Verfahren



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
137

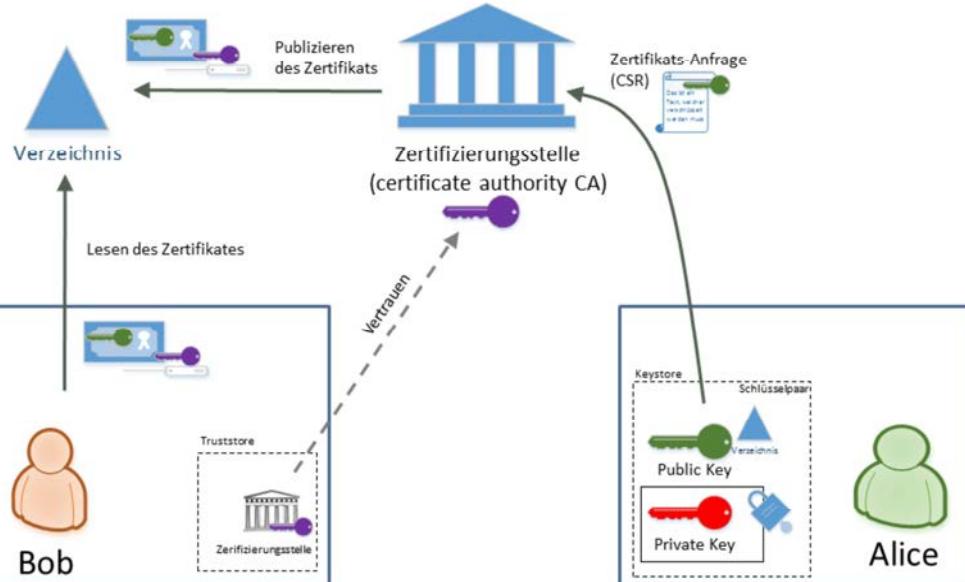
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Postfach

- Kombination beider Verschlüsselungsverfahren
 - Daten werden mit symmetrischen Schlüssel verschlüsselt
 - Der symmetrische Schlüssel wird mit Public Key des Empfängers verschlüsselt
- Vorteil:
 - Schlüsselverteilungsproblem ist gelöst und die Verschlüsselung kann trotzdem sehr schnell durchgeführt werden
- Hybride Verfahren:

Der Nachteil der asymmetrischen Verschlüsselung ist der hohe Rechenaufwand. Deswegen wird oftmals eine Kombination aus symmetrischer und asymmetrischer Verschlüsselung genutzt. Dabei wird eine Nachricht durch den Empfänger zunächst mit einem speziellen geheimen Schlüssel (Session Key) symmetrisch verschlüsselt. Anschließend wird dieser Schlüssel mit dem öffentlichen Schlüssel des Empfängers asymmetrisch verschlüsselt und übertragen. Der Empfänger kann nun asymmetrisch mit seinem privaten Schlüssel den Session Key und somit die eigentliche Nachricht symmetrisch entschlüsseln. Da nur der symmetrische Schlüssel verschlüsselt wird, bleibt der Rechenaufwand bei der asymmetrischen Verschlüsselung relativ gering.

Java EE 7 Security – Transportsicherheit (SSL / TLS)



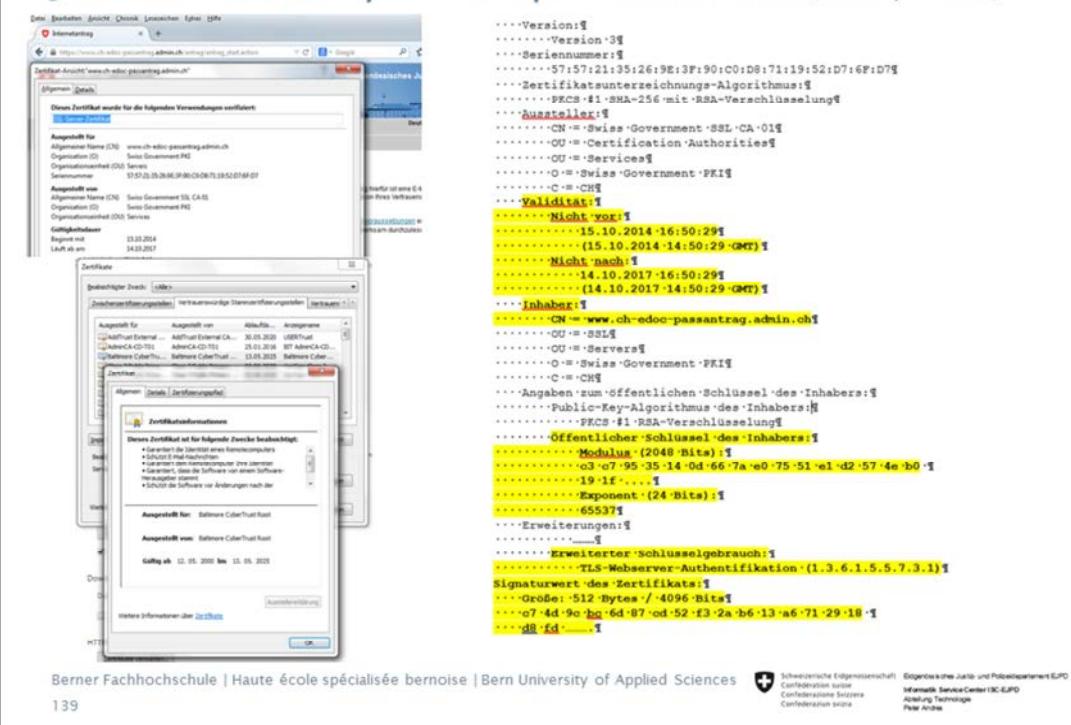
Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
138

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Wie wissen wir jedoch, dass der öffentliche Schlüssel wirklich Sender A gehört (siehe letzte Folie)?
- Dazu brauchen wir eine «trusted third authority» (TTP), im Umfeld der PKI (public key infrastructure) ist das eine «certificate authority» (CA), welcher wird vertrauen.
- Eine CA kann (mit entsprechenden Prozessen) einen öffentlichen Schlüssel «bestätigen». Sie tut dies, indem sie den öffentlichen Schlüssel und weitere Attribute wiederum mit ihrem privaten Schlüssel unterschreibt (signiert).
- Dem Zertifikat der CA (Root Zertifikat), mit welchem das Zertifikat signiert wurde, vertrauen wir (siehe z.B. im Browser unter «Vertrauenswürde Stammzertifizierungsstellen»).

Java EE 7 Security - Transportsicherheit (SSL / TLS)



The screenshot shows a web browser window with the URL <https://www.ch-edoc-passantrag.admin.ch/>. The browser displays a certificate chain. The top certificate is for `www.ch-edoc-passantrag.admin.ch` (Swiss Government SSL CA), issued by `Swiss Government PKI`. This certificate is signed by the `Baltimore CyberTrust Root`. The bottom certificate is for `Baltimore CyberTrust Root`, issued by `Baltimore CyberTrust`. The browser interface includes tabs for 'Zertifikat-Ansicht' and 'Zertifikat-Detailliert'. The right side of the screen shows the X.509 certificate structure in XML format, highlighting various fields like Version, Serial Number, and Public Key.

```
-----  
----- Version: 3  
----- Seriennummer: 5757252135269E3F90C0D8711952D76F07D7  
----- Zertifikatsunterzeichnungs-Algorithmus: RSA  
----- PFX: #1-SHA-256-mit-RSA-Verschlüsselung  
----- Aussteller: Swiss Government PKI  
----- CN = Swiss Government SSL CA-01  
----- OU = Certification Authorities  
----- OU = Services  
----- O = Swiss Government PKI  
----- C = CH  
----- Validität:  
----- Nicht vor: 15.10.2014-16:50:29  
----- (15.10.2014-14:50:29+0100)  
----- Nicht nach: 14.10.2017-16:50:29  
----- (14.10.2017-14:50:29+0100)  
----- Inhaber: CN = www.ch-edoc-passantrag.admin.ch  
----- OU = SSL  
----- OU = Servers  
----- O = Swiss Government PKI  
----- C = CH  
----- Angaben zum öffentlichen Schlüssel des Inhabers:  
----- Public-Key-Algorithmus des Inhabers: RSA  
----- PFX: #1-RSA-Verschlüsselung  
----- Öffentlicher-Schlüssel-des-Inhabers:  
----- Modulus (2048 Bits): 03:07:95:35:14:0d:66:7a:e0:75:51:e1:d2:57:4e:b0  
----- 19:1f:...  
----- Exponent (24-Bits): 65537  
----- Erweiterungen:  
----- Erweiterter-Schlüsselgebrauch: TLS-Webserver-Authentifikation (1.3.6.1.5.5.7.3.1)  
----- Signaturwert des Zertifikats:  
----- Größe: 512 Bytes / 4096 Bits  
----- 07:4d:9c:9b:64:97:cd:52:f3:2a:b6:13:a6:71:29:18  
----- d8:fd:...  
-----
```

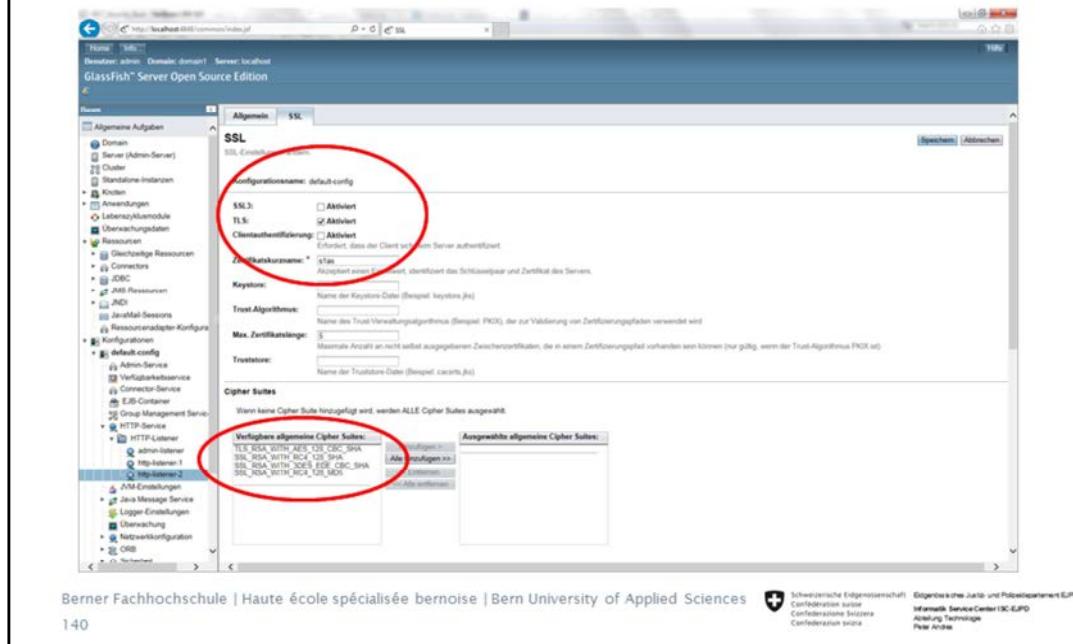
Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
139

 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederación Suiza
Confederazione Svizzera
Confederación Suiza
Confédération Suisse

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik ServiceCenter (ISC-JPO)
Abteilung Technik
Postfach

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Umsetzung im Glassfish (am Beispiel des Standard-Zertifikates «s1as»)



The screenshot shows the Glassfish Admin Console with the 'SSL' configuration page for the 's1as' certificate. The 'SSL' tab is active. Two specific sections are highlighted with red circles:

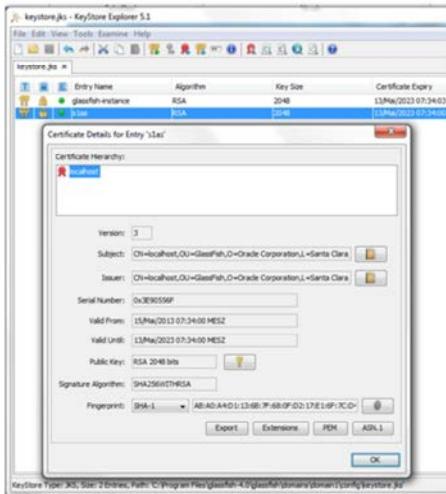
- Zertifikatname:** The field contains the value 's1as', which is highlighted with a red circle.
- Verfügbarer allgemeine Cipher Suites:** A list of cipher suites is shown, including:
 - SSL_RSA_WITH_AES_128_CBC_SHA
 - SSL_RSA_WITH_AES_256_CBC_SHA
 - SSL_RSA_WITH_3DES_EDE_CBC_SHA
 - SSL_DHE_RSA_WITH_AES_128_CBC_SHA

Below these sections, there are other configuration fields and tabs for 'Allgemein' and 'TLS'.

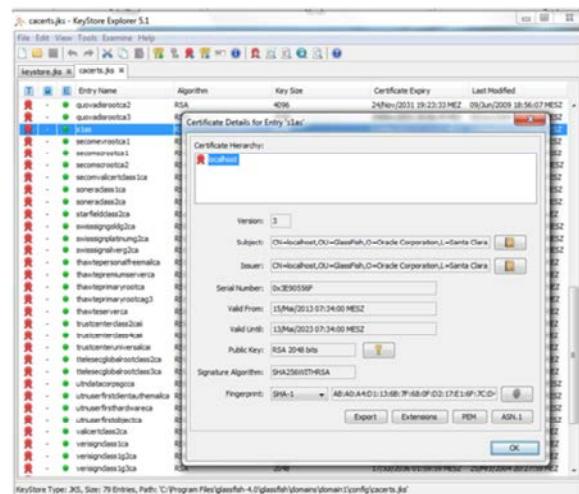
Java EE 7 Security – Transportsicherheit (SSL / TLS)

Umsetzung im Glassfish (am Beispiel des Standard-Zertifikates «s1as»)

Glassfish Keystore



Glassfish Truststore



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

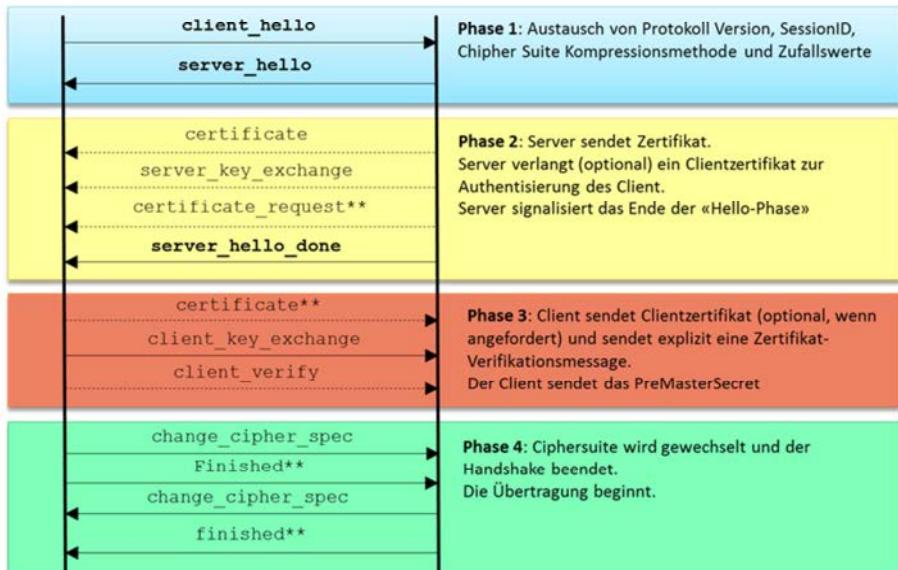
141

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Postfach

Java EE 7 Security – Transportsicherheit (SSL / TLS)

SSL-Handshake Protokoll



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
142

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

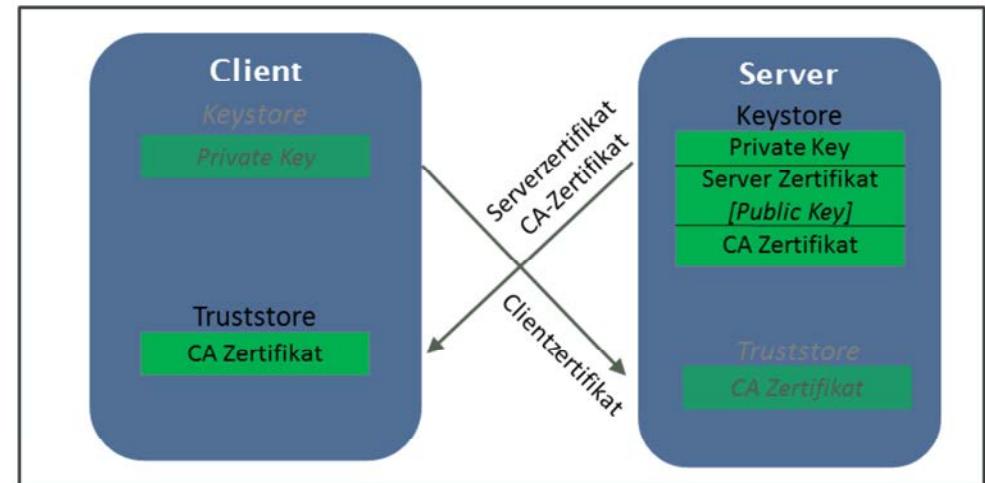
Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik ServiceCenter (ISC-JPO)
Abteilung Technologie
Peter Andri

- Das Ziel dieses Handshakes ist zum Einen die Authentisierung des Servers (optional auch des Clients) und zum Anderen die Frage, welches kryptographische Verfahren verwendet werden soll.
 - Das Ergebnis ist dann der Aufbau der endgültigen Verbindung. Dabei werden Public-Key Verfahren verwendet, um den symmetrischen Verbindungsschlüssel zu generieren.
- Der Client sendet seine Anfrage an den Server, dass er über SSL mit ihm kommunizieren möchte. Dabei sendet er unter anderem die Versionsnummer des Protokolls, das er verwenden möchte, eine Session ID und zufällig erzeugte Daten, um später einen Angriff erkennen zu können.
 - Der Server, der diese Anfrage erhält, sendet seinerseits die Daten, mit der er eine Verbindung aufbauen möchte, an den Client. Dazu gehören wieder die Protokoll-Version, die Session ID, Cipher Suite, (optional) Komprimierungsmethode und Zufallswerte.
 - Nun sendet der Server sein Zertifikat. Dieses Zertifikat beinhaltet unter anderem auch den öffentlichen Schlüssel des Servers, mit dessen Hilfe der Client seine Nachrichten verschlüsseln kann.
 - In einer weiteren Nachricht, der sogenannten *Certificate_Request* Nachricht, kann der Server nun optional auch die Authentisierung des Clients verlangen, wenn es sich bei der angefragten Informationsressource um einen Bereich handelt, der dies verlangt.
 - Nach diesen Nachrichten ist das *Server_Hello* abgeschlossen und der Server teilt das dem Client mit einer *Server_Hello_Done* Nachricht mit.

6. Wenn der Client diese Nachricht erhält, sendet er, wenn es verlangt wurde, sein Zertifikat zu seiner Authentisierung.
7. Nun generiert der Client unter Verwendung aller bisher ausgetauschten Daten das sogenannte *Premaster Secret* und sendet dieses verschlüsselt mit dem öffentlichen Schlüssel des Servers in der *Client_Key_Exchange*-Nachricht an den Server.
8. Wenn die Authentisierung des Servers (und optional des Clients) erfolgreich war, erzeugen nun beide Kommunikationspartner mit Hilfe des Premaster Secret und der vorher übertragenen Daten das *Master Secret*. Der Server muss dazu das Premaster Secret mit Hilfe seines privaten Schlüssels entschlüsseln. Dieses Master Secret dient zur Erzeugung des einmaligen (symmetrischen) Sitzungsschlüssels (*Session Key*).
9. Beide Partner melden das Beenden des Handshakes, indem sie auf die nun ausgehandelte symmetrische Verschlüsselung umschalten und eine *Finished*-Nachricht senden.
10. Die SSL-Übertragung kann beginnen. Client und Server verwenden den symmetrischen Sitzungsschlüssel, um den gegenseitigen Datenverkehr zu verschlüsseln und um die Integrität der übertragenen Daten zu gewährleisten.

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Keystore und Truststore



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
143



Eidgenössische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement EJPO
Informatik Service Center ISC-EJPO
Abteilung Technologie
Peter Andri

- Der private Schlüssel des Webservers muss sehr gut geschützt sein – idealerweise in einem HSM (Hardware Security Module)
- Das CA-Zertifikat, welche das Serverzertifikat ausgestellt hat, muss im Truststore des Clients sein.

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Deklarative Konfiguration im Web-Container (web.xml)

Listing: security-constraint im web.xml

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>SecureServlet</web-resource-name>
    <description>Matches all pages from protected</description>
    <url-pattern>/protected/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>user</role-name>
    <role-name>administrator</role-name>
  </auth-constraint>
  <user-data-constraint>
    <description>SSL required</description>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

Definition der Transport-Sicherheit

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
144



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

- Bei der Transportsicherheit (im allgemeinen SSL) kann gewählt werden zwischen NONE, CONFIDENTIAL und INTEGRAL, wobei CONFIDENTIAL und INTEGRAL von den JEE Applikations-Server identisch behandelt werden.
 - NONE = Keine verschlüsselte Verbindung nötig
 - CONFIDENTIAL = verschlüsselte Verbindung zwingend
 - INTEGRAL = verschlüsselte Verbindung zwingend
- Transportsicherheit garantiert Vertraulichkeit (durch Verschlüsselung) und Datenintegrität (durch Verschlüsselung).
- Mit der Einstellung «<transport-guarantee>CONFIDENTIAL</transport-guarantee>» muss der Zugriff über SSL erfolgen, ansonsten wird der Zugriff verweigert.
- Siehe auch <https://docs.oracle.com/cd/E19776-01/820-4502/beaqm/index.html>
- SSL kann auch Authentizität garantieren (für Clients mittels Authentisierung durch Client-Zertifikaten). Dazu wird im Login-Verfahren im web.xml «CLIENT-CERT» definiert.
- Wird nur der Webserver mit Hilfe von Zertifikaten Authentisiert spricht man von «one

way SSL». Werden der Webclient als auch der Webserver mit Hilfe von Zertifikaten authentisiert spricht man von «two way SSL»

- Transportsicherheit kann auch auf Stufe Applikations-Server forciert werden

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Deklarative (proprietäre) Konfiguration im EJB-Container (glassfish-ejb-jar.xml)

Listing: Konfiguration im glassfish-ejb-jar.xml

```
<enterprise-beans>
  <ejb>
    <ejb-name>SecureWebService</ejb-name>
    <webservice-endpoint>
      <port-component-name>Service</port-component-name>
      <login-config>
        <auth-method>BASIC</auth-method>
        <realm>SecureRealm</realm>
      </login-config>
      <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </webservice-endpoint>
  </ejb>
```

Definition der Transport-Sicherheit

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
145



Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

- Definition von SSL für einen Webservice im proprietären Deploymentdeskriptor von Glassfish 4

Java EE 7 Security – Transportsicherheit (SSL / TLS)

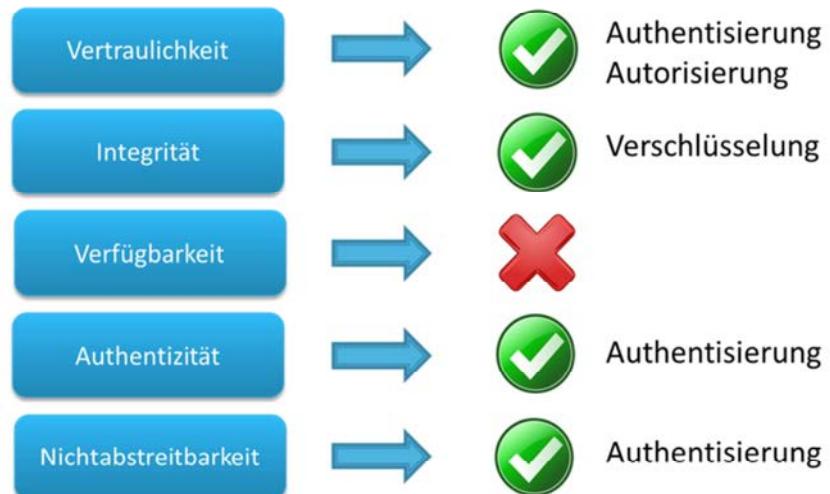
Pitfalls im Umgang mit SSL

- Wenn einmal auf SSL [HTTPs] gewechselt wurde, darf nicht mehr auf HTTP umgeschaltet werden können (Cookies können gesniffed werden).
- Die Standardeinstellungen im Applikationsserver müssen zwingend überprüft werden auf:
 - Aktualität und Support der neusten Protokolle (TLS 1.2)
 - Aktualität der Cipher-Suites
- Keystores müssen gut gesichert sein - am Besten in einem HSM (Hardware-Security-Modul)

Eine gute Empfehlung, welche Protokolle zu unterstützen sind und welche cipher-suites anzuwenden sind, ist hier zu finden:

https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2_pdf.pdf?__blob=publicationFile

Java EE 7 Security – Transportsicherheit (SSL / TLS)



Java EE 7 Security – Transportsicherheit (SSL / TLS)

Einschalten von SSL für die ganze (Web-) Applikation

- Importieren des «JEE7_Security_Kurs_Uebung_3_Loesung.zip» Projektes in Netbeans
- Erstellen einer <security-constraint> für
 - Beliebige Benutzer
 - Alle Methoden
 - Alle Ressourcen innerhalb der Web-Applikation (URL Pattern = «/*»)
- Die Lösung befindet sich im Projekt «JEE7_Security_Kurs_Uebung_4_Loesung.zip»
- **Zusatzfrage:** Gibt es mehrere Möglichkeiten?

7.2

Transportsicherheit

Zusatzaufgaben (Erstellen von Schlüsselpaaren und Zertifikaten)

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
149



Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Erstellen von Schlüsselpaaren und eines Zertifikates (selfsigned) für einen WebServer

- "%JAVA_HOME%/bin/keytool.exe" -genkey -alias WebServerCertificate -keyalg RSA -keypass changeit -storepass changeit -keystore "%GLASSFISH_HOME%/glassfish/domains/domain1/config/keystore.jks"

- Wie lautet Ihr Vor- und Nachname? [localhost]: [localhost]
- Wie lautet der Name Ihrer organisatorischen Einheit? [BFH]
- Wie lautet der Name Ihrer Organisation? [SWS]
- Wie lautet der Name Ihrer Stadt oder Gemeinde? [Bern]
- Wie lautet der Name Ihres Bundeslands? [Unknown]: [CH]
- Wie lautet der Ländercode (zwei Buchstaben) für diese Einheit? [CH]: [CH]

- In den Java Keystores werden Schlüssel paarweise erzeugt und mit einem „Alias“ versehen, damit man diese auch unterscheiden kann und wieder findet.
- Java bietet dazu ein Tool namens «keytool»
- Mit diesem Tool können Schlüsselpaare und Zertifikate (selfsigned oder auch signing-requests CSR) erstellt werden.
- Siehe auch
<https://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html>
- Ein nützliches grafisches Tool ist der KeyStore-Explorer, siehe auch <http://keystore-explorer.sourceforge.net/>
- changeit ist das Default-Password der Java-Key und Trust-Stores. Es muss im produktiven Umfeld gewechselt werden.

Java EE 7 Security – Transportsicherheit (SSL / TLS)

- ▶ **Export des WebServer-Zertifikates**
- ▶ `"%JAVA_HOME%\bin\keytool.exe" -export -alias WebServerCertificate -storepass changeit -file c:\temp\webserver.cer -keystore "%GLASSFISH_HOME%\glassfish\domains\domain1\config\keystore.jks"`
- ▶ **Erstellen eines Certificate-Signing Request**
- ▶ `"%JAVA_HOME%\bin\keytool.exe" -certreq -keyalg RSA -alias WebServerCertificate -file c:\temp\certreq.txt -keystore "%GLASSFISH_HOME%\glassfish\domains\domain1\config\keystore.jks"`
- ▶ **Versenden des Certificate-Signing Request an eine CA**
- ▶ An dieser Stelle stellt eine autorisierte CA ein entsprechendes Server-Zertifikat aus

- Man sieht, dass das Zertifikat «self-signed» ist (keine Hierarchie).
- Der CN muss dabei den Hostnamen des Webservers enthalten (hier localhost). Dies wird vom Browser geprüft und bei Nichtübereinstimmen wird er Browser eine Fehlermeldung generieren oder den Zugriff unterbinden. Normalerweise ist hier der Name der Servers (Host).
- Der command «keytool -**export**» dient zum Exportieren des Server-Zertifikates (nur öffentlicher Schlüssel) in das File `webserver.cer`
- Es wäre ebenfalls möglich einen CSR für eine CA zu machen – und ein Zertifikat von einer offiziellen CA zu bekommen (siehe command «keytool -**certreq**»)

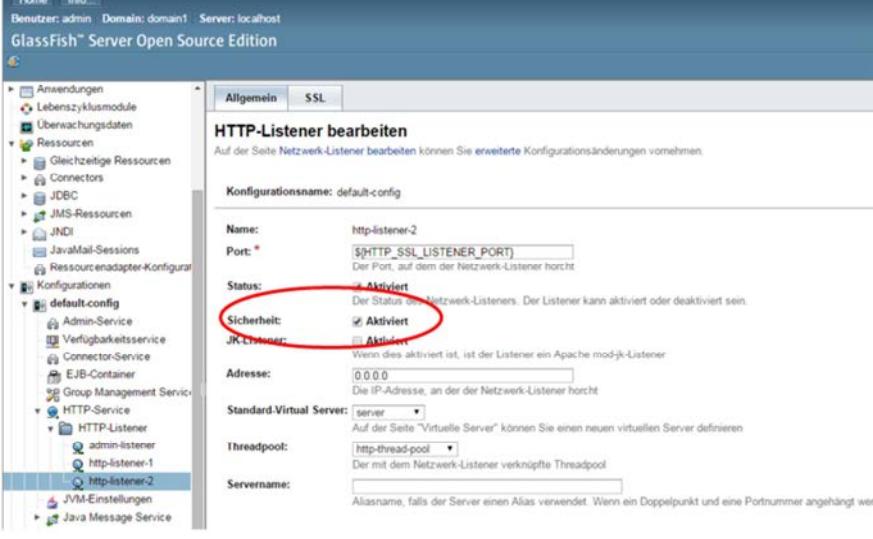
Java EE 7 Security – Transportsicherheit (SSL / TLS)

- **Import des (Root) WebServer-Zertifikates (z.B. ausgestellt von der CA)**
- "%JAVA_HOME%\bin\keytool.exe" -import -v -trustcacerts -alias WebServerCertificate -file c:\Temp\webserverCA.pem -keystore "%GLASSFISH_HOME%\glassfish\domains\domain1\config\cacerts.jks" -keypass changeit -storepass changeit

- In der SSL-Konfiguration im Glassfish-Applikationsserver kann beim HTTPS Listener der Alias des entsprechenden Zertifikates angegeben werden.
- Es muss sichergestellt sein, dass der Browser das Root-Zertifikat importiert hat (Truststore), oder das Zertifikat von einer bereits vertrauten CA ausgestellt worden ist. Sonst wird der Browser den Zugriff verhindern oder eine Warnung ausstellen
- Im Glassfish kann nun im HTTPS-Listener der neue Alias für das neue Webserver-Zertifikat angegeben werden. (Konfigurationen->server-config->HTTP-Service->HTTP-Listener->http-listener-2 «Tab SSL», «Zertifikatskurzname»)

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Konfiguration von SSL im Glassfish 4.x



The screenshot shows the Glassfish 4.x Administration Console. The left sidebar navigation tree includes 'Anwendungen', 'Lebenszyklusmodule', 'Überwachungsdaten', 'Ressourcen' (with 'Gleichzeitige Ressourcen', 'Connectors', 'JDBC', 'JMS-Ressourcen', 'JNDI', 'JavaMail-Sessions', and 'Ressourcenadapter-Konfiguration'), 'Konfigurationen' (with 'default.config' containing 'Admin-Service', 'Verfügbarkeitservice', 'Connector-Service', 'EJB-Container', 'Group Management Service', 'HTTP-Service' with 'admin-listener', 'http-listener-1', and 'http-listener-2'), 'JVM-Einstellungen', and 'Java Message Service'. The 'SSL' tab is selected in the top navigation bar. The main content area is titled 'HTTP-Listener bearbeiten' and shows configuration for 'http-listener-2'. The configuration includes:

- Name:** http-listener-2
- Port:** `$HTTP_SSL_LISTENER_PORT` (The port on which the network listener listens.)
- Status:** Aktiviert (Activated)
- Sicherheit:** Aktiviert (Activated)
- JK-Listener:** Aktiviert (Activated) (If activated, it is an Apache mod-jk listener.)
- Adresse:** 0.0.0.0 (The IP address on which the network listener listens.)
- Standard-Virtual Server:** server (The virtual server associated with the network listener.)
- Threadpool:** http-thread-pool (The thread pool associated with the network listener.)
- Servername:** (Alias name for the server if it uses an alias.)

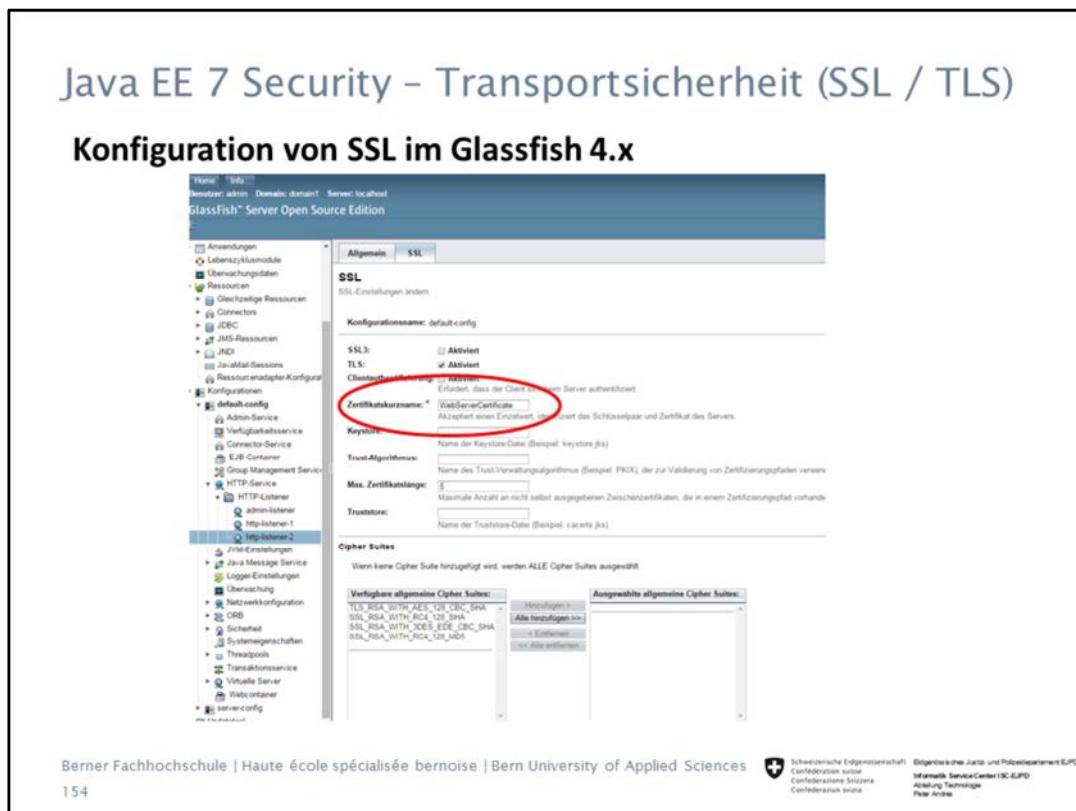
The 'Sicherheit' (Security) checkbox is circled in red.

Page footer: Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences | 153

Page footer: Schweizerische Eidgenossenschaft | Confederation of Switzerland | Confédération Suisse | Confederazione Svizzera | Confederación Suiza | Espace Culturel et Sportif | Service Center ITC-EPO | Abteilung Technik | Postfach

Java EE 7 Security – Transportsicherheit (SSL / TLS)

Konfiguration von SSL im Glassfish 4.x



The screenshot shows the Glassfish 4.x Administration Console with the 'SSL' tab selected. The 'Zertifikatskennname' field is highlighted with a red circle. The configuration details are as follows:

- SSL:** Aktiviert
- TLS:** Aktiviert
- ClientAuth:** MANDATORIEN
- Zertifikatskennname:** * Erfordert, dass der Client den eigenen Server authentifiziert.
- Keystore:**
- TrustAlgorithm:** Maximale Anzahl an nicht selbst ausgewiesenen Zwischenzertifikaten, die in einem Zertifizierungspfad vorhanden sind.
- Truststore:**

Cipher Suites: Wenn keine Cipher Suite hinzugefügt wird, werden ALLE Cipher Suites ausgewählt.

Verfügbare allgemeine Cipher Suites:	Ausgewählte allgemeine Cipher Suites:
<small>TLS_RSA_WITH_AES_128_CBC_SHA -> TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS_RSA_WITH_RC4_128_MD5</small>	<small>Hinzufügen > <input type="checkbox"/> Alle hinzufügen >> -> Entfernen -> Alle entfernen</small>

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
154

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technik
Postfach

Besten Dank – viel Erfolg im Umgang mit Security



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
155

 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
© 2013 Scott Adams, Inc. / Dist. by Universal Uclick
9-6-13

Eidgenössisches Justiz- und Polizeidepartement (EJPD)
Informatik Service Center (ISC-EJPD)
Abteilung Technologie
Peter Andri

- Webservices im Web-Tier können gemäss Kapitel 4 «ganz normal» abgesichert werden

9

Der Bookstore

- Webservices im Web-Tier können gemäss Kapitel 4 «ganz normal» abgesichert werden

Java EE 7 Security -Übung Bookstore: Ziele

► **Vertraulichkeit**

- Users des Bookstore müssen authentisiert und autorisiert werden, um gewisse Funktionen auszuführen und Daten einzusehen

► **Integrität**

- Die Kommunikation ist zu schützen

► **Security-Roles**

- Erstellen eines **Rollen und Berechtigungskonzeptes** für den Administrator, die Customer und die Employees.
- Der Admin kann Employees erfassen (Admin GUI)
- Employees können Kunden verwalten (REST-API)
- Kunden können sich registrieren und ihre Daten (Accounts, Orders) verwalten (REST-API)
- Alle können Bücher suchen und den shopping cart verwenden (ohne Authentisierung)

157

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences



Eigentum des Justiz- und Polizeidepartement (JPO)
Informatik Service Center (ISC-JPO)
Abteilung Technologie
Peter Andri

Java EE 7 Security -Übung Bookstore: Design

- Erstellen eines Security Design für den Bookstore anhand des Berechtigungskonzeptes
- Aufzeigen im Komponenten-Modell wo bereits Security eingebaut worden sind (nicht JEE7 Security) und wo man noch Security Aspekte einbauen muss

