Aufgabe 5 - RESTful Webservice mit Java - Security-Aspekte

Ziel

Das Ziel dieser Aufgabe ist der Erwerb von Kompetenzen in der Absicherung von RESTful Web Services. Dazu gehört das grundsätzliche Verständnis von Methoden der Authentifizierung und Autorisierung. Darüberhinaus zielt diese Aufgabe auf das erweiterte Verständnis der Funktionsweise von Protokollen (*HTTP*) auf Ebene 7 (*application layer*) des ISO-OSI-Schichtenmodells ab. Ebenso soll ein Bewusstsein für die Komplexität von Sicherheitsthematiken geschaffen werden, welche alle (Software-)Systemkomponenten durchdringen.

Aufgabe

Die Aufgabe besteht in der Erweiterung des in der letzten Aufgabe erstellten PPS um Funktionalitäten der Authentifizierung und der Autorisierung/Zugriffskontrolle. Hierzu soll das Spring Security Framework (oder ein anderes, geeignetes Framework) verwendet werden. Ein Zugriff auf die angebotenen Funktionalitäten der REST-Endpoints soll nur mehr durch authentifizierte Clients, welche auch die entsprechende Autorisierung besitzen, möglich sein. Die Authentifizierung der Clients soll mittels eines Benutzernamens und eines Passwortes erfolgen, welche, z.B., bei den einzelnen Requests übergeben werden (Basic HTTP Authentication[1]).

Passwörter können auf Seiten des PPS/der REST-Services in einer simplen in-memory Datenbank gespeichert werden, jedoch muss dies in einer sicheren Art und Weise erfolgen. Dies bedeutet, dass Passwörter nicht als plain text gespeichert werden sollen, sondern nur deren (idealerweise mit einem Salt versehenen) Hash-Werte, welche mittels einer geeigneten Hash-Funktion[2] (z.B.: bcrypt) erstellt wurden. Zur Unterstützung soll hierfür ein simples Programm[3] erstellt werden, welches mittels Benutzereingabe über die Konsole ein Passswort (und etwaige zusätzliche Konfigurationsparameter, abhängig von der gewählten Hash-Funktion) erhält, das eingegebene Passwort mittels einer geigneten Hash-Funktion hasht und dann den erzeugten Hash-Wert in der Konsole ausgibt.

Zur Autorisierung der Clients soll es zwei verschiedene Rollen geben: Worker und Production Manager. Die im Rahmen der vorangehenden Aufgabe erstellten Funktionalitäten sind nun

[1] Basic Authentication garantiert hierbei nicht die Vertraulichkeit der übertragenen Zugangsdaten, hierfür wird für gewöhnlich HTTPS verwendet. Für diese Aufgabe muss dieser Aspekt fürs Erste nicht berücksichtigt werden.

[2] Die Anforderungen an Hash-Funktionen für Passwörter sind speziell, da hier im Vergleich zu anderen kryptographischen Hash-Funktion eine hohe Effizienz des Algorithmus keine wünschenswerte Eigenschaft darstellt (da dies Brute-Force-Angriffe und die Generierung von Rainbow-Tables erleichtert). Deshalb existieren speziell für diesen Zweck entwickelte Funktionen.

[3] Dieses Programm ist separat vom PPS zu sehen, es dient nur der Unterstützung bei der Erstellung der nötigen Daten, um in Folge die implementierten Authentifizierungs- und Autorisierungsfunktionalitäten zu testen.

wie folgt aufgelistet für die beiden Rollen zugänglich:

- Alle Fertigungsaufträge zurückgeben (Worker, Production Manager)
- Alle Fertigungsaufträge sortiert nach ihrem Rang zurückgeben (Worker, Production Manager)
- Neuen Fertigungsauftrag anlegen, und hinten in der Reihenfolge anhängen (Production Manager)
- Neuen Fertigungsauftrag anlegen, und vorne in der Reihenfolge einfügen (Production Manager)
- Neuen Fertigungsauftrag anlegen, und an bestimmter Position in der Reihenfolge einfügen (Production Manager)
- Maschinenzuordnung eines Fertigungsauftrags ändern (Worker, Production Manager)
- Rang eines Fertigungsauftrags ändern (Worker, Production Manager)
- Einen Fertigungsauftrag löschen (Production Manager)
- Alle Fertigungsaufträge löschen (Production Manager)

Zum Testen der implementierten Mechanismen sollen zwei User mitsamt der relevanten Daten (Benutzername, Passwort-Hash, Rolle) angelegt und in der in-memory Datenbank hinzugefügt werden: Ein User hat die Rolle Worker und der andere User hat die Rolle Production Manager.

Wie bereits bekannt, verwendet das HTTP-Protokoll Statuscodes. Die für die Aufgabe relevanten Codes sind 401 Unauthorized (der Request wurde nicht erfüllt, da kein valider Berechtigungsnachweis erbracht wurde - erfolgreiche Authentifizierung würde Zugriff ermöglichen) und 403 Forbidden (der Request ist korrekt, jedoch weigert sich der Server diesen zu gewähren, z.B., aufgrund fehlender Zugriffsrechte). Zur Veranschaulichung müssen mittels eines Werkzeugs wie Postman für jede Funktionalität drei verschiedene Requests abgesetzt werden: Ein Request ohne jegliche Authentifizierung des Clients, ein Request mit der Authentifizierung als ein User mit der Rolle Worker und ein Request mit der Authentifizierung als ein User mit der Rolle Production Manager. Die einzelnen Requests und deren Ergebnisse sollen mittels Screenshots dokumentiert werden.

Im Rahmen der letzten Aufgabe wurde ebenfalls ein einfacher Client mittels Java HTTP Client entwickelt, um eine Benutzerkonsole der Maschinen zu simulieren. Damit dieser Client weiterhin auf die nun abgesicherten REST-Services zugreifen kann, muss er entsprechend erweitert werden.

Als Basis für die gesamte Implementierung dient die Ausarbeitung der letzten Aufgabe.

Hinweis: Das Spring Security Framework enthält eingebaute Schutzmechanismen gegen Cross-Site Request Forgery (CSRF), wodurch bei jedem HTTP-Request von einem Client, welcher Daten verändern kann (POST, PUT, DELETE), z.B. ein entsprechendes Token mitgesendet werden muss, da sonst der Request mit dem Statuscode 403 Forbidden

abgelehnt wird. Für diese Aufgabe können die Schutzmechanismen gegen CSRF deaktiviert werden.

Bewertung

Folgenden Mindestanforderungen muss diese Aufgabe genügen:

- Ein Zugriff auf die angebotenen Funktionalitäten der REST-Services ist für nicht authentifizierte Clients nicht möglich (401 unauthorized)
- Authentifizierte Clients können nur auf die für die Rolle ihres Users zugelassenen Funktionalitäten (wie oben angeführt) zugreifen (403 forbidden für nicht autorisierte Funktionalitäten)
- Client, Server und Passwort-Hasher sind Maven-Projekte

Um die volle Punktezahl zu erreichen, müssen folgende Punkte umgesetzt sein:

- Die angeführten Mindestanforderungen wurden umgesetzt
- Es werden auf Seiten des Servers nur die Passwort-Hashes der User gespeichert (und es wurde eine dafür vorgesehene Hash-Funktion verwendet)
- Das erstellte Passwort-Hasher-Programm kann verwendet werden, um für Passwörter Hash-Werte zu generieren (und es wurde eine dafür vorgesehene Hash-Funktion verwendet)
- Der im Rahmen von Aufgabe 4 erstellte Client kann weiterhin auf die verwendeten Funktionalitäten der Services zugreifen

Dokumentation & Abgabe

Abzugeben ist eine einzelne Archivdatei (.zip) mit folgendem Inhalt:

- Exportiertes Projekt (.zip) oder exportierte Projekte (wenn Client, Server und Passwort-Hasher in separaten Projekten umgesetzt wurden). Es sind nur Textdateien erlaubt; binäre Dateien wie beispielsweise Java Bibliotheken, das `target`-Unterverzeichnis oder das kompilierte Projekt dürfen nicht in der Archivdatei enthalten sein.[4]
- Screenshots vom Zugriff auf die RESTful Webservices mittels einem Werkzeug

Deadline:

- Deadline: Dienstag, 29. November 2022, 23:59 Uhr
- Format: Archivdatei im Format .zip
- Abgabe via Moodle

[4] Am besten `mvn clean` vor dem Erstellen des Zip-Archivs der Projekte ausführen.

Startpunkt der Recherche

- Beschreibung des Szenarios
- Ausarbeitung von Aufgabe 4
- TB1 - Sichere Kommunikation in verteilten Systemen
- TB3 - REST
- Übungsbeispiel
- https://docs.spring.io/spring-security/reference/servlet/authorization/authorize-http-requests.html
- https://docs.spring.io/spring-security/reference/6.0.0-M7/servlet/authentication/passwords/in-memory.html#page-title
- https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCryptPasswordEncoder.html
- https://openjdk.java.net/groups/net/httpclient/intro.html
- https://docs.oracle.com/en/java/javase/11/docs/api/java.net.http/java/net/http/HttpClient.html
- http://zetcode.com/java/httpclient/
- https://dzone.com/articles/java-11-standardized-http-client-api
- https://en.wikipedia.org/wiki/Cross-site_request_forgery
- https://en.wikipedia.org/wiki/Basic_access_authentication
- https://en.wikipedia.org/wiki/Bcrypt