Hochschule Esslingen
University of Applied Sciences

Offensive Security

Projekt Bericht

im Studiengang
Softwaretechnik und Medieninformatik

vorgelegt von

Jan Binder
Matr.-Nr.: 749707

am 07. Februar 2019
an der Hochschule Esslingen

Prüfer/in: Thomas Fischer

Inhaltsverzeichnis

Json Web Tokens

1.1 What is it

1.1.1 Overview

Json Web Tokens are Access Tokens that can be used to authenticate Users against a (web-)application. As you can already tell by the name, they are based on a json object and are mainly used to check if a user is who he pretends to be and if the data that was send by this user remains unchanged.

1.1.2 Structure

Json Web Tokens consist of three parts:

- Header
- Payload
- Signature

These parts get separated by dots to make it easy to recognize where which part starts. To understand how each part works, it is necessary to know how each one is build.

The splitting of JWT does not end with the Header, Payload and Signature. These Parts are also split into smaller parts (Two to be exact). The first one is the algorithm, that is used to sign the Token (“alg”: “HS256”). This could be a HMAC SHA256 algorithm or an RSA.

```
Example:
{
  "alg": "HS256",
  "typ": "JWT"
}
```

The algorithm is then followed by the “typ”- Field. This field always has the same Value(“JWT”) to show that this Object is a Json Web Token.

The next part of the JWT is the Payload which contains the claims that are transported with the token. These claims are Information about the User like username or if the user has admin rights. Some of these claims are registered and can be used to provide helpful information. Additionally, there are a lot of Public claims as well. These can be freely chosen but should be defined by the IANA (Internet Assigned Numbers Authority). Finally, the user can define its own claims that are not standardized.

```
Example:
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

The last part of the JWT is now the signature. Its structure is defined by the JSON Web Signature. To create the signature, the header and the Payload must be Base64 encoded and separated by a dot. Afterwards this string gets hashed with the Function that was mentioned in the Header and the result gets Base64 encoded as well. Now we take all the Encoded Parts of the Token and concatenate them with a dot in-between.

```
jwt = base64(header) + "." + base64(payload) + "." + base64(hash)
```

1.1.3 How does it work

When logging in the user will get an JWT in the Response of his Authorization-Request. This JWT will then be sent with every Request of the User. This typically happens in the Authorization-Header of the Http-Request, but it can also be sent within the Cookie-Header or in the URL as a Parameter that must be analyzed by the respective application as shown in the examples below.

Authorization-Header:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Cookie-Header:

```
Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Url-Parameter:

```
http://example.com/path?jwt_token=eyJhbGciOi-
JIUzI1NiIsInR5cCI6IkpXVCJ9...
```

If the user tries to access a route that needs authentication, the application checks if the User has a valid JWT Token and if this is the case, he can access the requested resource.

1.2 Pros and Cons

1.2.1 Disadvantages

- With signed tokens there is only guaranteed that the information of the user is unchanged, but everyone can read the information that is contained in it, so keep in mind to not send any critical information about any subject in the token
- If anybody gets access to the token he can act as the user and gets access just as the user would have
- If the token is used as stateless session token it is hard to revoke the tokens validity so if you want to lock out certain users because of strange behavior, for example accessing the application from multiple IP- Addresses you can’t do much about it.

1.2.2 Advantages

- JWT can be used on a lot off different platforms like Node.js, Python, Java and many more that can be found at https://jwt.io/. This means it can easily be used for Single Sign On
- JWT can easily be built and analyzed
- JWT can be used for Stateless Sessions. This means there is no need to store the token in a database
- It has a very small overhead

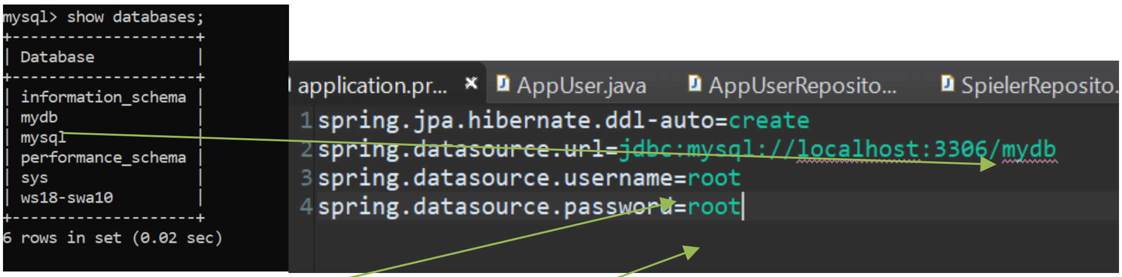
1.3 Example Applikation

The Application is not perfect yet. I documented the link to the Git repo where you can find it but I may use another project we did this semester and change the authentication to JWT this project would be at https://github.com/BiNd3r94/DropboxClone . With an explanation on how to use it in the Readme file.

1.3.1 How to find and use it

The example App can be taken from Github. You can just clone it from https://github.com/BiNd3r94/HandballApp. To use it you need a Mysql database, Maven and a browser of your choice (I used Firefox).

After you did download/clone the Repository you need to configure your Mysql database and the “application*.properties”-Files to enable the Access of the Application to the database



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mydb               |
| mysql              |
| performance_schema |
| sys                |
| ws18-swa10         |
+--------------------+
6 rows in set (0.02 sec)
```

```
spring.jpa.hibernate.ddl-auto=create
spring.datasource.url=jdbc:mysql://localhost:3306/mydb
spring.datasource.username=root
spring.datasource.password=root
```

Database-Username Database-Password

The database you want to use in the MySql must be set up before the first start of the Application but the "create" Statement makes sure that the tables are created at the first startup. After the first start it can be set to update so the data won't be destroyed everytime.

To start the Application you can simply type *mvnw spring-boot:run* under windows or *./mvnw spring-boot:run* under Linux.

After the Application did start you can browse to *localhost:8080/* and will be automatically redirected to the login page where you need to authenticate yourself.

To do that you have to first add a user by signing him up at *localhost:8080/users/sign-up*

Now you can sign in to the application with the new user.

2 Quellen

https://de.wikipedia.org/wiki/JSON_Web_Token

https://jwt.io/introduction/

JWT Handbook (Ebook)

Ehrenwörtliche Erklärung

Name:	Binder	Vorname:	Jan
Matrikel-Nr.:	749707	Studiengang:	SWB

Hiermit versichere ich, Jan Binder, dass ich den vorliegenden Praxissemesterbericht mit dem Titel „Continuous Delivery und der DevOps Ansatz“ selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebene Literatur und Hilfsmittel verwendet habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Asperg, 26.09.2017

Ort, Datum

Unterschrift