REST API Security Authentication and Certificates

Guadalupe Ortiz Bellot

Department of Computer Science and Engineering

Universidad de Cádiz

Contents

Contents

REST Security Importance

Contents

2. XML Deployment Descriptor Definitions

- **Security Constraints** – We use them define access **permissions** to the resources defined in our API **resource collection**. To protect our resources we will define resource collections as **URL patterns and HTTP Methods.**
- **Authorization Constraints** – They indicate which **users** and using which **roles** are **permitted to access** to a particular resource collection previously defined.
- **Login Configuration** – It is used to identify the **authentication method** which will be used to access to the restricted resources. We also have to identify the *Realm* in which the user will be authenticated.
- **Security Roles** – They define **which roles will be used for the permission to access** a particular set of resources in the API.

See https://avaldes.com/jax-rs-security-using-basic-authentication-and-authorization/

Contents

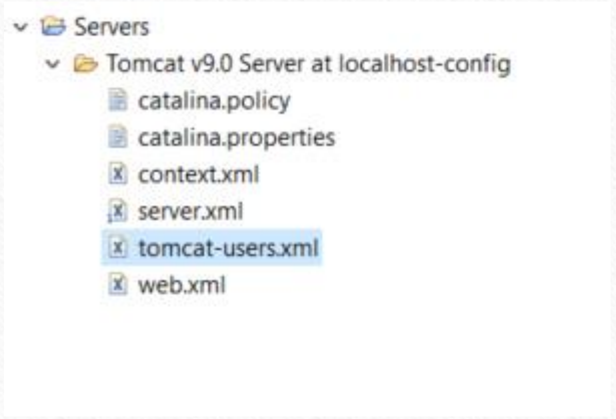
3. Basic Authentication Introduction

- Most used techniques in all types of applications:
- username + password
- Credential validation

- Main drawback: credentials are propagated in a plain way from the client to the server.

3. Basic Authentication
Creating a new user in Tomcat

- Edit tomcat-users.xml in the tomcat in Eclipse
- Double click on tomcat-users.xml to open it



- Add a new role if necessary

```
<role rolename="basicRestUser"/>
```

- Add a new user

```
<user username= "restUser1" password="restUser1passwd" roles="basicRestUser"/>
```

- Restart the server

3. Basic Authentication Modifying web-xml (i)

```
<security-constraint>
  <display-name>Secure REST Area</display-name>
  <web-resource-collection>
    <web-resource-name> Hello REST</web-resource-name>
   <url-pattern>/*</url-pattern>
    <http-method>PUT</http-method>
    <http-method>POST</http-method>
    <http-method>DELETE</http-method>
  </web-resource-collection>
[…]
```

These are the resources affected by the following security constraints

3. Basic Authentication Modifying web-xml (ii)

```
[…]
 <auth-constraint>
    <role-name>basicRestUser</role-name>
  </auth-constraint>
      <user-data-constraint>
        <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
</security-constraint>
[…]
```

These are the authorization constraints (role with access) and the transport protocol security specifications

Are you allowed?

3. Basic Authentication Modifying web-xml (iii)

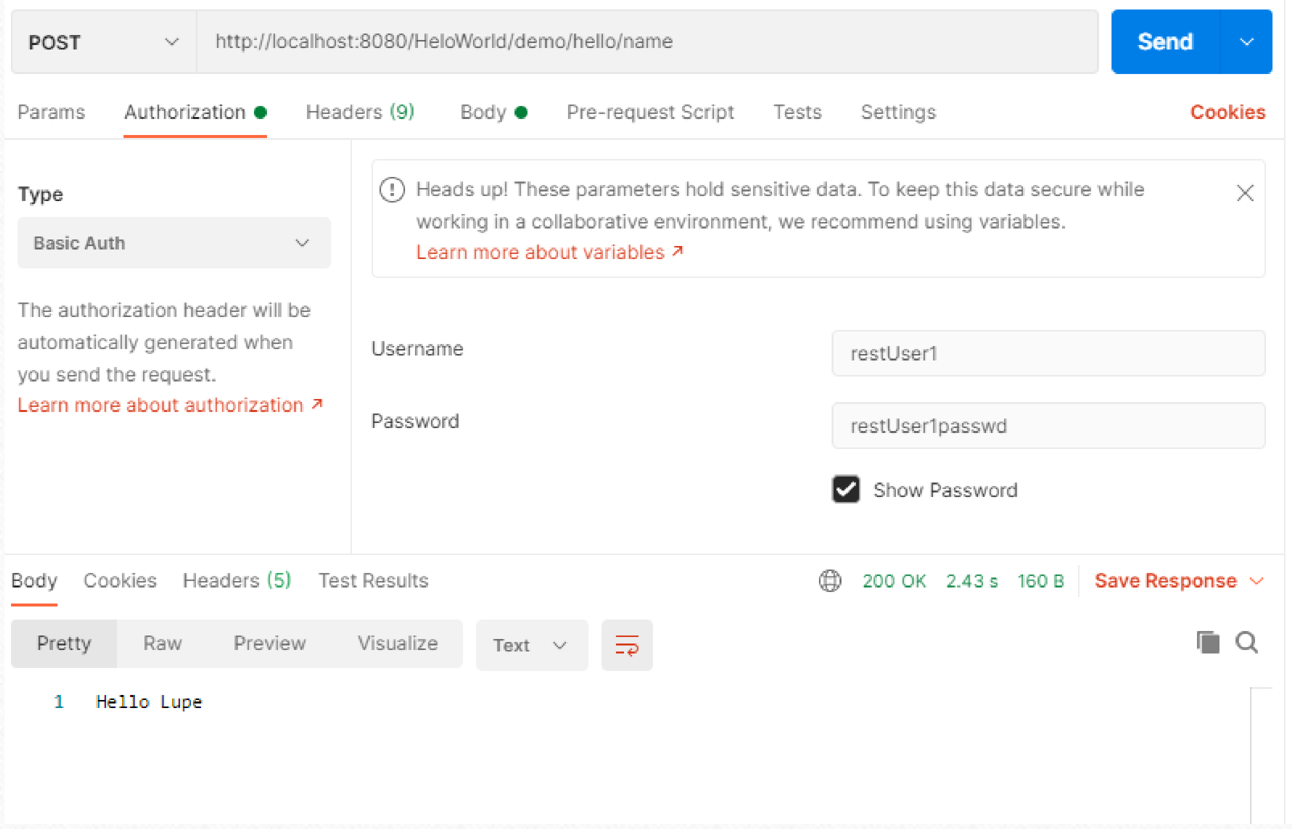
```
[…]
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>default</realm-name>
</login-config>
<security-role>
  <role-name>basicRestUser</role-name>
</security-role>
```

These are the authentication method required and the roles accepted

Are you who you say you are?

3. Basic Authentication Testing it from Postman

- **You have to restart the server**
- If we submit the request without authentication we will receive a 401 error.
- Click on Authorization and add Basic Authentication
- Do not forget to include the body.



http://localhost:8080/HeloWorld/demo/hello/name

Type: Basic Auth

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.
Learn more about variables ↗

Username: restUser1

Password: restUser1passwd

☑ Show Password

200 OK 2.43 s 160 B

```
Hello Lupe
```

Contents

4. Digest Authentication
Introduction

- It uses a hash function to encrypt the password

- Digest md5 authentication applies a function on the combination of the values of the username, realm and password.

4. Digest Authentication
Configuring Digest Authentication

Configuring Digest Authentication:

- Enable HTTP Digest Authentication in our **web.xml** file
- Update tomcat users
- Update server.xml (if necessary)

4. Digest Authentication
Set HTTP Digestion in the web.xml file (i)

```
<security-constraint>
<display-name>Secure REST </display-name>
<web-resource-collection>
<web-resource-name>Hello REST Service</web-resource-name>
<url-pattern>/*</url-pattern>
<http-method>PUT</http-method>
<http-method>POST</http-method>
<http-method>DELETE</http-method>
</web-resource-collection>
```

These are the resources affected by the following security constraints

4. Digest Authentication
Set HTTP Digestion in the web.xml file (ii)

```
<auth-constraint>
<role-name>digestRestUser</role-name>
</auth-constraint>
</security-constraint>
```

These are the authorization constraints (role with access) and the transport protocol security specifications

4. Digest Authentication
Set HTTP Digestion in the web.xml file (iii)

```
<login-config>
<auth-method>DIGEST</auth-method>
<realm-name>UserDatabase</realm-name>
</login-config>

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

These are the authentication method required and the roles accepted

4. Digest Authentication
Update Tomcat users

- Edit tomcat-users.xml in the tomcat in Eclipse

- Add a new role if necessary

  <role rolename="digestRestUser"/>

- Add a new user

  <user username="restUser2" password= "restUser2passwd" roles=" digestRestUser"/>

- Restart the server (or later if you are doing more changes in the server)

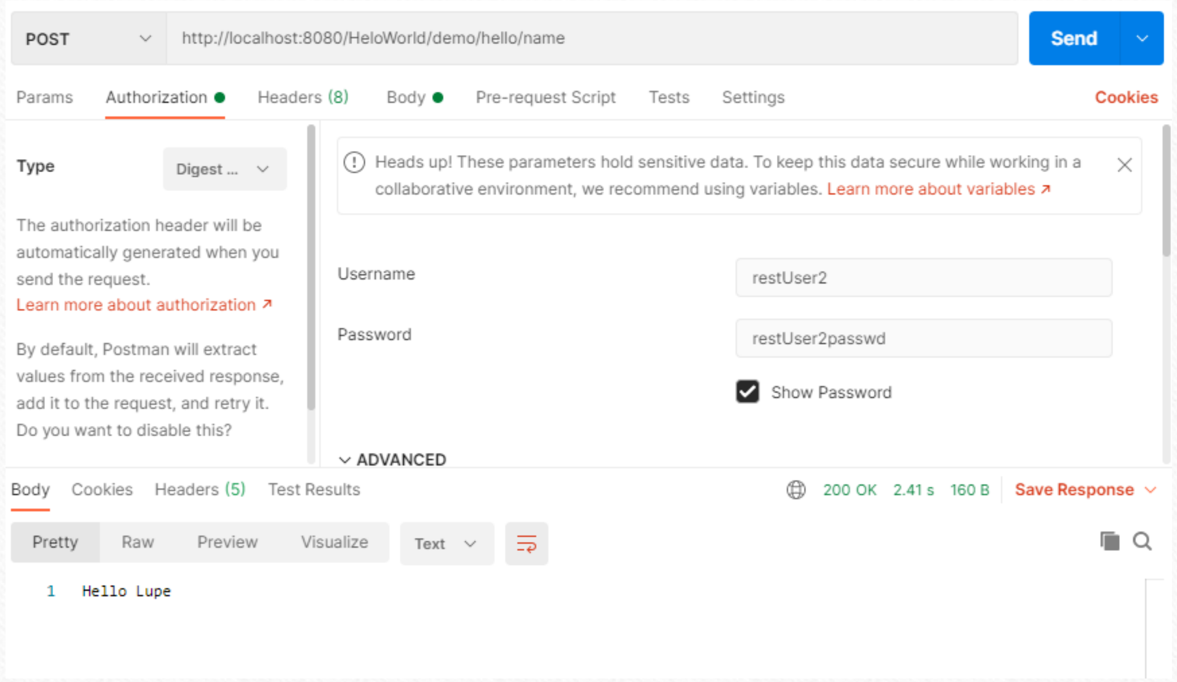
4. Digest Authentication Server.Xml by default

By default we store the password in the tomcat-users.xml in clear text, but we could select an external database and additional encryption mechanism. Check that the following code is in your server.xml:

```
<Realm className="org.apache.catalina.realm.LockOutRealm">

<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/>    </Realm>


 <GlobalNamingResources>

<Resource auth="Container" description="User database that can be updated and
saved" factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
name="UserDatabase" pathname="conf/tomcat-users.xml"
type="org.apache.catalina.UserDatabase"/>  </GlobalNamingResources>
```

4. Digest Authentication Testing it from Postman

- **You have to restart the server**
- If we submit the request without authentication we will receive a 401 error.
- Click on Authorization and add Digest Authentication
- Do not forget to include the body.



POST http://localhost:8080/HeloWorld/demo/hello/name

Type: Digest

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

By default, Postman will extract values from the received response, add it to the request, and retry it. Do you want to disable this?

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Username: restUser2

Password: restUser2passwd

☑ Show Password

200 OK 2.41 s 160 B

```
Hello Lupe
```

Contents

5. Securing HTTP Methods

- **We have to bear in mind that when HTTP methods are included** within a constraint definition, the protections defined by the **constraint** are applied only to those **methods**.
- **If HTTP methods are not listed** within a constraint definition, then the protections defined by the **constraint** will apply to the **complete set of HTTP methods**.

See https://docs.oracle.com/javaee/6/tutorial/doc/gmmku.html

- You can secure your REST API by including the methods and/or the URIs in your constraints, using the combination that better adapts to your needs.

See https://docs.oracle.com/javaee/6/tutorial/doc/gmmku.html

Contents

5. Using Certificates
Introduction

- Trust agreement is established between the server and the client through certificates
- They must be signed by an agency, which is known as CA
- To test it we can generate our own certificates

5. Using Certificates
Generating the certificate

- Generate the certificate
- keytool -genkeypair -alias *username* -keyalg RSA -keypass *password* -storepass *password* -keystore C:\Development\mykeystore

- keytool -genkeypair -alias *restUser* -keyalg RSA -keypass *restUserPasswd* -storepass *restUserPasswd* -keystore C:\Development\mykeystore

- **Note that the keypass and storepass passwords should be same.** Otherwise, you have to add additional information to Tomcat configuration files or you get the following error “java.io.IOException: Cannot recover key“.

5. Using Certificates
Verifying the certificate

- Verify if the certificate is created properly by this command:
- keytool -list -keystore C:\Development\mykeystore
- You should obtain a digital fingerprint (SHA1):

```
ED:54:B5:2C:F5:9B:13:48:AC:07:91:A0:F1:0F:4D:F0:9B:AB:B3:21:F7:CA:4B:44:61:5B:4C:F1:FD:B4:9A:91
```

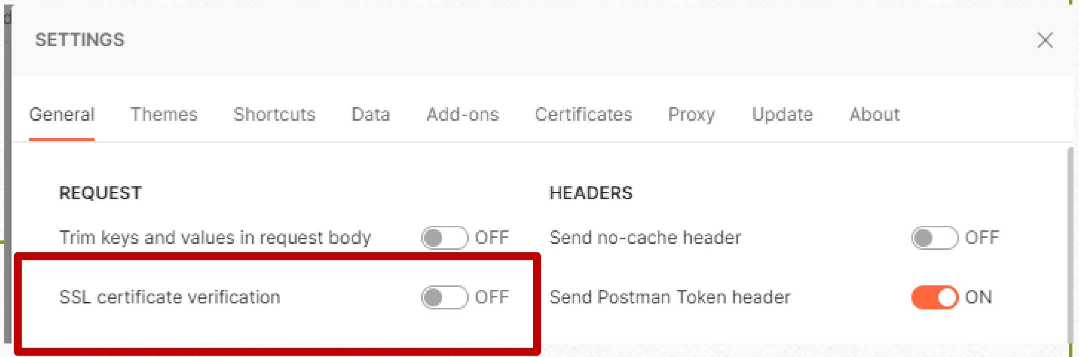
5. Using Certificates
Configuring server.xml in Tomcat

- Uncomment and complete in server.xml:

```
    <Connector
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443"
        maxThreads="200"
        scheme="https" secure="true" SSLEnabled="true"
        keystoreFile="C:\Development\mykeystore"
        keystorePass="restUserPasswd"
        clientAuth="false" sslProtocol="TLS"
  />
```

5. Using Certificates Testing it

- **By default Autosigned certificates are not allowed in Postman**
- (Postman) File→Settings→General → SSL Certificate Verification OFF
- Test it with basic authentication
- You simply have to invoke through https and port 8443



REQUEST

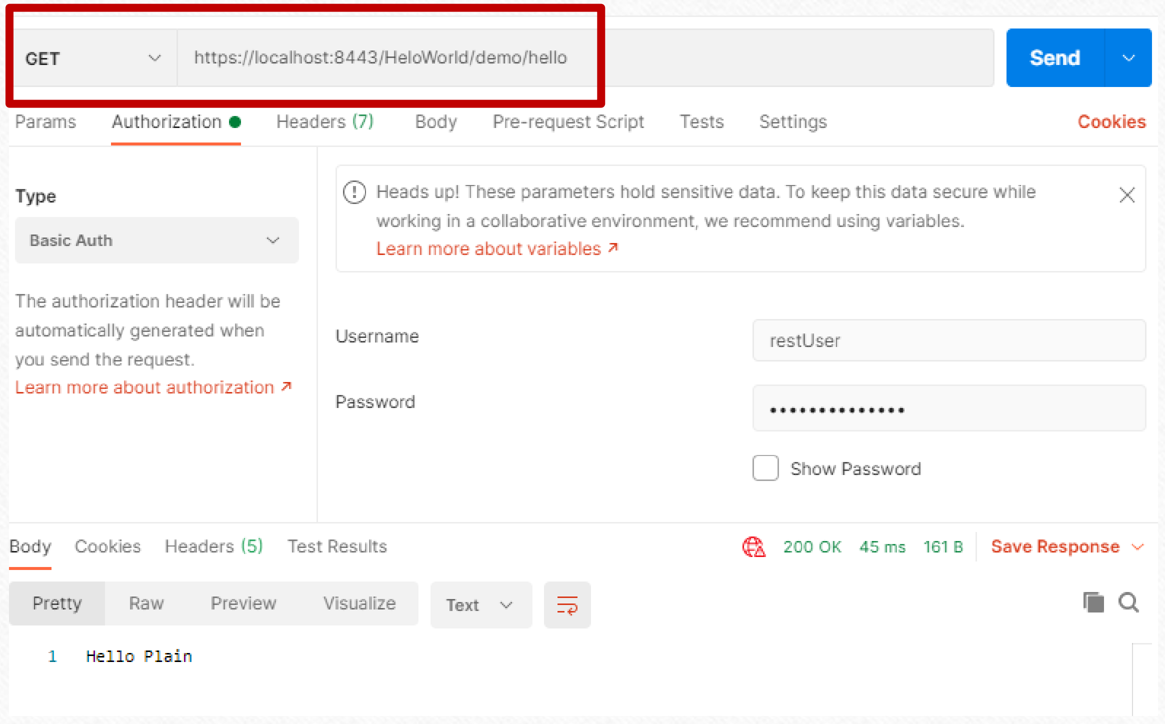
Trim keys and values in request body OFF

SSL certificate verification OFF

HEADERS

Send no-cache header OFF

Send Postman Token header ON



GET https://localhost:8443/HeloWorld/demo/hello

Type: Basic Auth

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.
Learn more about variables ↗

Username: restUser

Password: •••••••••••••••

200 OK 45 ms 161 B

```
Hello Plain
```

5. Using Certificates Configuring web.xml

- **You can force it** in the web.xml:

```
[…]
 <auth-constraint>
    <role-name>basicRestUser</role-name>
  </auth-constraint>
      <user-data-constraint>
       <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
[…]
```

Support Bibliography and References

- Amauri Valdés, Developers Corner. https://avaldes.com/category/java-development/jax-rs/
- Andrés Salazar C., René Enríquez. RESTFUL Java Web Services Security. https://www.safaribooksonline.com/library/view/restful-java-web/9781783980109/
- Bill Burke. RESTFUL Java with JAX-RS 2.0. https://www.safaribooksonline.com/library/view/restful-java-with/9781449361433/
- Oracle. The Java EE 6 Tutorial. Secured HTTP Resources. https://docs.oracle.com/javaee/6/tutorial/doc/gmmku.html