**Security REST APIs with Spring Security and JWT Tutorial**

This tutorial demonstrates how to secure a REST API using JSON Web Tokens (JWT) and Spring Security, based on the GitHub repository [spring-boot-rest-api-auth-jwt-tutorial](https://github.com/givanthak/spring-boot-rest-api-auth-jwt-tutorial).

The Representational State Transfer Application Programming Interface (REST API) facilitates communication and data exchange between systems over the internet. REST APIs utilize the JavaScript Object Notation (JSON) data format and rely on standard HTTP methods such as GET, POST, PUT, and DELETE.

By adhering to these common HTTP methods and status codes, REST APIs provide a consistent interface across various applications while supporting all content types. The REST API architecture promotes stateless communication between clients, such as users, and servers. This design does not store information about current sessions or connections. Instead, applications manage access, resource manipulation, and session handling independently.

Because REST APIs regulate data transmission between applications, they are a common target for attackers seeking unauthorized access to sensitive data. Implementing access control mechanisms is crucial to safeguarding your REST API. This involves verifying all clients requesting access by implementing two key processes:

* **Authentication**: Confirming that the client is genuinely who it claims to be.
* **Authorization**: Ensuring the client is permitted to perform the requested actions.

A robust way to secure REST APIs is through the use of JSON Web Tokens (JWTs). These are digitally signed JSON objects that can be verified and trusted. They enable:

* **Authentication**: Using public/private key pairs to confirm that the signature-holder is legitimate.
* **Authorization**: Including an access token in every request after a successful login to control user actions effectively.

Before starting, ensure you have the following tools and knowledge:

* **Java** (JDK 8 or later).
* **Spring Boot** and **Spring Security** basics.
* **Maven** for building the project.
* **Postman** or cURL for testing the API.

1. **Setting up the Project**

Clone the repository to your machine, Build the project using Maven to install the necessary dependencies ($ **mvn clean install**), then start the SpringBoot application. The server will start on <http://localhost:8080>.

1. **JWT (JSON Web Token)** is a compact, URL-safe token that represents claims between two parties. It is widely used for securing REST APIs. A JWT typically consists of three parts:

* **Header**: Contains the type of token (JWT) and the signing algorithm.
* **Payload**: Holds the claims (e.g., user ID, roles).
* **Signature**: Ensures the token's integrity and authenticity.

Workflow

* **Login**: The client sends a login request with credentials.
* **Token Issuance**: If authentication is successful, the server generates a JWT and sends it to the client.
* **Token Usage**: The client includes the JWT in the Authorization header for subsequent requests.
* **Validation**: The server validates the JWT before processing the request.

1. **Exploring the Codebase**

**User Model:** Defines the user entity

**Token Generation**

O imagine care conține text, captură de ecran, Font

Descriere generată automat

**JWT filters** in JWTAuthenticationFilter.java class:

**O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat**

**WebSecurity class:** Configures security settings for the application

1. Testing the API

O imagine care conține text, captură de ecran, Font, document

Descriere generată automatThe app defines following CRUD APIs:

O imagine care conține text, captură de ecran, Font

Descriere generată automat

For the working example app, reference this git project: