Authentication, Authorization, and PKI in a Spring Boot Project

1. Introduction

In any modern web application, securing access to resources is critical. This involves not only identifying and verifying users (authentication) but also ensuring they have the necessary permissions to access specific resources (authorization). Additionally, Public Key Infrastructure (PKI) provides a robust framework for securing communications and verifying identities through cryptographic methods. This document explains how these concepts are implemented in my Spring Boot project.

2. Authentication

2.1 What is Authentication?

Authentication is the process of verifying the identity of a user or system. It ensures that the entity requesting access to a resource is who they claim to be. This process typically involves validating credentials such as usernames, passwords, tokens, or certificates.

2.2 Authentication Methods in the Project

In my Spring Boot project, multiple authentication methods are implemented:

- **Basic HTTP Authentication:** This method involves sending a base64-encoded string containing the username and password with each HTTP request. It is straightforward but should be used over HTTPS to avoid exposing credentials.
- **Form-Based Authentication:** Users log in through a custom HTML form. Upon submission, credentials are sent to the server for verification, and if successful, the user is granted access.
- **JWT (JSON Web Token) Authentication:** After a successful login, a JWT is generated and returned to the client. This token is then used for subsequent requests, allowing the server to authenticate users without needing to store session information.

2.3 Implementation in the Project

- **Basic HTTP Authentication:** Configured in Spring Security settings to handle requests requiring basic authentication.
- **Form-Based Authentication:** Managed through a custom login controller that handles user credentials and forwards authenticated users to their destination.
- **JWT Authentication:** Implemented using a custom JwtRequestFilter that intercepts HTTP requests, validates the JWT, and sets the authentication context.

3. Authorization

3.1 What is Authorization?

Authorization determines whether an authenticated user has permission to access a specific resource or perform an action. While authentication verifies identity, authorization controls access.

3.2 Role-Based Access Control (RBAC) in the Project

I implemented Role-Based Access Control (RBAC), where users are assigned specific roles, such as USER, ADMIN. Access to resources is granted based on these roles.

3.3 Implementation in the Project

- **Role-Based Access Control:** Configured in Spring Security settings, where specific URLs or actions are restricted to users with certain roles. For example, only users with the ADMIN role can access the /api/users/create endpoint.

4. Public Key Infrastructure (PKI)

4.1 What is PKI?

Public Key Infrastructure (PKI) is a framework for managing digital keys and certificates that enable secure communication, data encryption, and identity verification. PKI uses pairs of cryptographic keys—public and private keys—where the public key encrypts data and the private key decrypts it.

4.2 PKI in Your Project

In the project, PKI is used to enhance authentication by issuing and validating digital certificates, ensuring that communication is secure and that entities are verified through cryptographic means.

4.3 Implementation of PKI

- **Certificate Generation:** I generated RSA keys and self-signed certificates using tools like OpenSSL.
- **HTTPS Configuration:** the server is configured to use HTTPS, which requires an SSL/TLS certificate. This ensures that all communications between clients and the server are encrypted.
- **Token-Based Authentication with PKI:** When a user authenticates via the /authenticate endpoint, a token is generated using RSA private keys. This token can then be validated with the corresponding public key, ensuring its authenticity.

5. Conclusion

By implementing these security mechanisms—authentication, authorization, and PKI—I have ensured that the Spring Boot application is robust, secure, and capable of handling sensitive information and user data. Authentication verifies user identity, authorization controls access, and PKI secures communication and identity verification through cryptographic means.

This multi-layered approach to security is critical in modern applications, providing the necessary safeguards to protect the application from unauthorized access and potential security threats.