Authentication, Authorization, and PKI Documentation

This document provides an overview of Authentication, Authorization, and Public Key Infrastructure (PKI) with a focus on their implementation in Spring Boot.

Authentication

Authentication is the process of verifying the identity of a user, system, or entity. In Spring Boot, authentication is typically handled by Spring Security, a powerful and highly customizable authentication and access-control framework, and can be implemented using various mechanisms, such as:

Basic Authentication: Basic Authentication is a simple authentication scheme built into the HTTP protocol. Examples include; Formed-base Authentication, In-Memory Authentication and JDBC Authentication. In-Memory Authentication is used primarily for development and testing purposes. User credentials are stored directly within the application's memory, while in JDBC Authentication, credentials are stored in a database, and Spring Security verifies them against the database records. Form-based authentication uses an HTML form to collect user credentials.

OAuth2 and OpenID Connect: These are more advanced authentication mechanisms used for integrating third-party identity providers like Google, Facebook, or custom OAuth2 servers.

JWT Authentication: JSON Web Tokens (JWT) are a popular method for implementing stateless authentication. JWT is a compact token format used to securely transmit information between parties.

In Spring Boot, 'AuthenticationManager' is a central component that orchestrates the authentication process.

Authorization

Authorization determines what authenticated users are allowed to do within the application. Authorization can also be defined as the process of determining whether a user has the right to access a resource or perform an action. In Spring Boot, this is typically done using roles and permissions.

Role-Based Access Control (RBAC): RBAC is a method of regulating access based on the roles of individual users. Users are assigned roles, and roles have permissions that define what actions the user can perform.

Method-Level Security: It allows you to apply security constraints directly on methods within your services or controllers by using annotations like, @PreAuthorize`, `@ PostAuthorize` `@Secured`, `@RolesAllowed`, `@PreFilter`, and `PostFileter`.

Public Key Infrastructure (PKI)

PKI is a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption. It can also be defined as a framework that uses cryptographic techniques to secure communications over a network. It involves the use of public and private key pairs and certificates.

Certificates and Certificate Authorities: Digital certificates are electronic documents used to prove the ownership of a public key. They are issued by Certificate Authorities (CAs), bind a public key to the identity of an individual or organization.

SSL/TLS in Spring Boot: Spring Boot applications commonly use TLS/SSL for secure communication. PKI is the foundation for the SSL/TLS protocols, ensuring the confidentiality and integrity of data transmitted over the network. To enable HTTPS in Spring Boot, you need

to configure SSL. First, generate a self-signed certificate or obtain one from a CA. Then, configure your application.properties.

Key Management: Proper management of keys and certificates is crucial. Spring Boot provides support for configuring keystores and truststores, which hold private keys and trusted certificates, respectively.