Authentication, Authorization, and PKI

Owner: NII NARTEY
Reviewer: THOMAS DARKO

Introduction

During my journey of learning Spring Security, I had some thinking about whether there are other developers who share a similar experience. Those who struggle to comprehend the inner workings of Spring Security components or those who seek to learn through demonstration code and aspire to develop their own projects based on that foundation, etc. Therefore, I made the decision to write this article, with the hope that it may be helpful in some way.

This article aims to showcase a basic web application with APIs secured by Spring Security. Instead of delving deep into introducing Spring Security concepts, as they are readily available on the official website, I will focus on providing comprehensive explanations while constructing project components. At the end of the article, you will find a link to the GitHub repository containing the functional code for reference.

Terms and Concepts

Spring Security does not secure networks or computers, but applications : it is involved in the dialogue between the application and the user (or between two applications). This dialogue is managed by the Spring *Dispatcher Servlet*, which redirects http requests to the controller classes of the application. Concisely, the role of Spring Security is to insert operations in this interaction, thanks to a bunch of *Servlet Filters*. This group of filters is the **Filter Chain** of Spring Security.

Filters

A filter is a fundamental notion in Spring Boot Security since it intercepts each request before it reaches the servlets.

We may create filters in any combination, and they will prevent the request from progressing to the next filter unless all of the requirements are fulfilled.

Familiarising ourselves with these notions will tremendously assist us in navigating the authentication and authorization setup.

The *filter chain* deals with 2 fundamental concepts :

- **authentication** : the user must be identified by a **username**/**password** combination.
- **authorizations** : users are not equal regarding the operations they are allowed to do. As an example, a user who is not an administrator should not be allowed to modify the account of another user.

These fundamentals are explained further below:

Authentication — *Who are you? Can you prove it?*

Authentication is the process of ascertaining whether or not someone or something is who or what they claim to be. There are several methods of authentication based on the credentials a user must supply when attempting to log in to an application. The first form is **knowledge-based authentication** (*KBA*), which is based on the individual's knowledge. We may place username-password authentication in this category. Following that is **possession-based authentication** (*PBA*), which is based on something the user possesses. In this case, phone/text messaging, key cards and badges, and so on can be utilised. The third type of authentication is **multi-factor authentication** (*MFA*), which is a mixture of the first two and requires the user to give two or more verification factors in order to obtain access to the application.

Authorization — *Can you do this?*

Many people mistakenly use the phrases authentication and authorization interchangeably, although they are not the same thing. Simply defined, authorization is the process of determining which precise **rights/privileges/resources** a person *has*, whereas authentication is the act of determining who someone *is*. As a result, authorization is the act of **granting someone permission** to do or have something. Furthermore, authorization is frequently viewed as both the preparatory setting up of permissions and the actual checking of permission values when a user is granted access.

Principal

A principal is a person who is authenticated through the authentication procedure. Consider it a **presently logged-in user** of the currently logged-in account. One person can have many IDs (for example, Google), but there is generally only one logged in user. You may get it from the security context, which is connected to the current thread and hence to the current request and its session. As a result, the Spring Security principle can only be accessed as an Object and must be cast to the appropriate UserDetails instance, as I will describe in more detail in the future blogs.

Granted Authority

These are the user's fine-grained permissions that define what the user may perform. Each conferred power can be thought of as an individual privilege. *READ_AUTHORITY* and *WRITE_AUTHORITY* are two examples. The name is arbitrary in this context, and we may alternatively refer to the idea of authority by using privilege.

Role

This is a coarser-grained set of authorities granted to the user (for example, *ROLE_TEACHER, ROLE_STUDENT…*). A role is expressed as a String that begins with "ROLE." The semantics we attach to how we utilise the feature is the primary distinction between granted authority and role.

What is public key infrastructure (PKI)?

Public key infrastructure is an important aspect of internet security. It is the set of technology and processes that make up a framework of encryption to protect and authenticate digital communications.

PKI uses cryptographic public keys that are connected to a digital certificate, which authenticates the device or user sending the digital communication. Digital certificates are issued by a trusted source, a certificate authority (CA), and act as a type of digital passport to ensure that the sender is who they say they are.

Public key infrastructure protects and authenticates communications between servers and users, such as between your website (hosted on your web server) and your clients (the user trying to connect through their browser. It can also be used for secure communications within an organisation to ensure that the messages are only visible to the sender and recipient, and they have not been tampered with in transit.

The main components of public key infrastructure include the following:

- Certificate authority (CA): The CA is a trusted entity that issues, stores, and signs the digital certificate. The CA signs the digital certificate with their own private key and then publishes the public key that can be accessed upon request.
- Registration authority (RA): The RA verifies the identity of the user or device requesting the digital certificate. This can be a third party, or the CA can also act as the RA.
- Certificate database: This database stores the digital certificate and its metadata, which includes how long the certificate is valid.
- Central directory: This is the secure location where the cryptographic keys are indexed and stored.
- Certificate management system: This is the system for managing the delivery of certificates as well as access to them.
- Certificate policy: This policy outlines the procedures of the PKI. It can be used by outsiders to determine the PKI's trustworthiness.

Understanding how PKI works

Public key infrastructure uses asymmetric encryption methods to ensure that messages remain private and also to authenticate the device or user sending the transmission.

Asymmetric encryption involves the use of a public and private key. A cryptographic key is a long string of bits used to encrypt data.

The public key is available to anyone who requests it and is issued by a trusted certificate authority. This public key verifies and authenticates the sender of the encrypted message.

The second component of a cryptographic key pair used in public key infrastructure is the private, or secret, key. This key is kept private by the recipient of the encrypted message and used to decrypt the transmission.

Complex algorithms are used to encrypt and decrypt public/private key pairs. The public key authenticates the sender of the digital message, while the private key ensures that only the recipient can open and read it.

PKI certificates

The core of a public key infrastructure is trust. It is important for a recipient entity to know without a doubt that the sender of the digital certificate is exactly who they claim to be.

Trusted third-party CAs can vouch for the sender and help to prove that they are indeed who they say they are. Digital certificates are used to verify digital identities.

Digital certificates are also called *PKI certificates* or *X.509 certificates*. A PKI certificate offers proof of identity to a requesting entity, which is verified by a third party and works like a digital passport or driver's license.

The PKI certificate will contain the following:

- Distinguished name (DN) of the owner
- Owner's public key
- Date of issuance
- Expiration date
- DN of the issuing CA
- Issuing CA's digital signature

Conclusion

Learning how to implement Spring Security is never finished : there is always something else to know. An exhaustive knowledge of the subject is nevertheless not required to provide decent security to your application. Here are the 3 bullet points you want to memorise, if nothing else :

➔ Spring Security is mostly a sequence of filters to use (or not) and to configure.

➔ Security provided by Spring Security is based on the authentication / authorization duo.

➔ The filters and URL protections are configured in the @EnableWebSecurity annotated class : that is the class to read in order to quickly understand how Spring Security is implemented in an application.

Sources :

- spring security architecture
- spring security reference documentation
- spring security reference expression language
- spring security reference headers
- Why Public Key Infrastructure Is a Good Idea. (March 2001). *Computer Weekly*.
- EJBCA Enterprise. (2022). PrimeKey AB.
- OpenSSL. (2021). The OpenSSL Project Authors.
- Cloudflare/CFSSL. (2022). GitHub, Inc.
- Xipki/xipki. (2022). GitHub, Inc.
- PKI Main Page. Dogtag PKI.