

• The Authentication Filter — Delegates the authentication request to the authentication

manager and, based on the response, configures the security context.

• The Authentication Manager — Uses the authentication provider to process the

authentication.

• The Authentication Provider—Implements the authentication logic.

• The User Details Service—Implements user management responsibility. The

authentication provider uses it in the authentication logic.

• The Password Encoder—Implements password management. The authentication

provider uses it in the authentication logic.

• The Security Context—Keeps the authentication data after the authentication process.

In the following paragraphs, I discuss the following auto-configured beans:

UserDetailsService & PasswordEncoder

An object that implements a UserDetailsService contract from Spring Security manages the details about the users.

The PasswordEncoder does two things:

• Encodes a password

• Verifies if the password matches an existing encoding

A PasswordEncoder exists together with the default UserDetailsService. When we replace the

default implementation of the UserDetailsService, we must also specify a PasswordEncoder.

Authentication method Spring chooses by default is the HTTP Basic access authentication.

Basic authentication only requires the client to send a username and a password through the HTTP Authorization header. In the value of the header, the client attaches the prefix Basic, followed by the Base64 encoding of the string that contains the username and password, separated by a colon (:).

***NOTE : HTTP Basic authentication doesn’t offer confidentiality of the credentials. Base64 is only an encoding method for the convenience of the transfer, not an encryption or hashing method. While in transit, if intercepted, the credentials can be seen.***

Vulnerabilities in authentication and authorization

*Authentication* represents the process

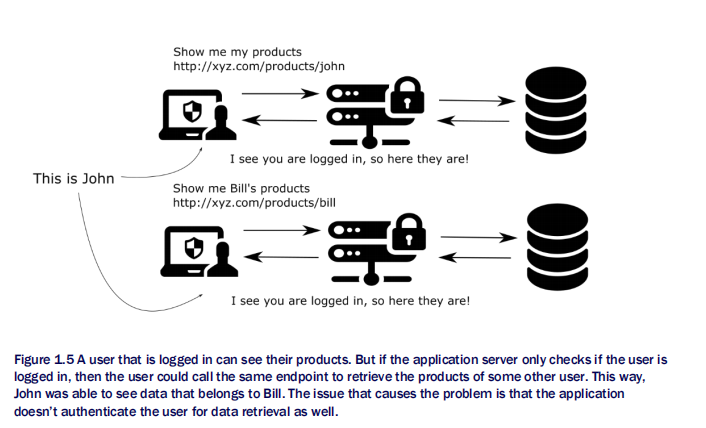
in which an application identifies someone trying to use it. When someone or something uses

the app, we want to find their identity so that further access is granted or not.

*Authorization* is the process of establishing if an authenticated caller has the privileges to

use specific functionality and data.

1. Broken Authorization :



1. session fixation

If present, it permits an attacker to impersonate a valid user by reusing a previously generated

session ID. This vulnerability could happen if, during the authentication process, the web

application does not assign a unique session ID, and this could make possible the reuse of

existing session IDs. Exploiting this vulnerability consists of obtaining a valid session ID and

making the intended victim’s browser use it.

The WebSecurityConfigurerAdapter class contains three different overloaded configure()

methods.

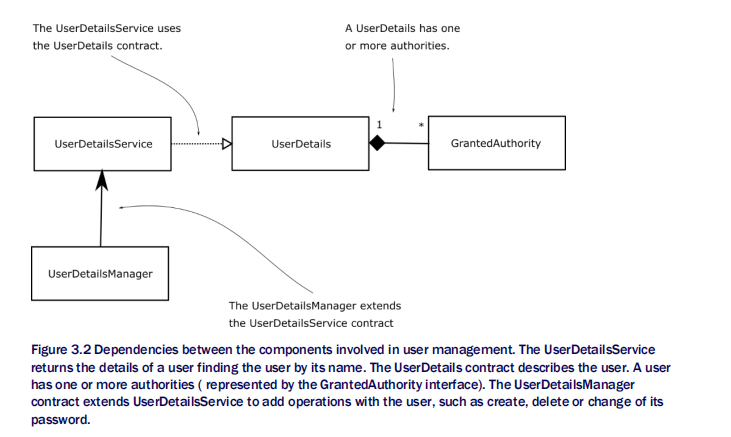
Configure(HttpSecurity http)

Configure(AuthenticationManagerBuilder auth)

The UserDetailsService is only responsible for retrieving the user by its

username.

The UserDetailsManager adds behavior that refers to adding, modifying, or deleting the user



For Spring Security, a user definition should respect the UserDetails contract. The UserDetails contract represents the user, as understood by Spring Security.

The User class from the org.springframework.security.core.userdetails package is a simple way to build instances of the UserDetails type. Using the class, you create immutable instances of UserDetails. You need to provide at least a username, and a password, and the username shouldn’t be the empty string.

To describe the authorities in Spring Security, you use the GrantedAuthority interface.

NOTE It is good practice to verify that the interface is marked as functional with the @FunctionalInterface annotation before implementing it with lambda expressions. The reason for this practice is that if the interface is not marked as functional, it could mean that its developers reserve the right to add more abstract methods to it in future versions. The GrantedAuthority interface is not marked as functional. In this book, we will implement it with lambda expressions often to make the code snippets shorter and easier to understand. However, I recommend you avoid doing so in your applications until it is explicitly marked as functional.