İbrahim AKIN

1. What are Authentication and Authorization?

Authentication is the process of verifying who someone is, whereas authorization is the process of verifying what specific applications, files, and data a user has access to. The situation is like that of an airline that needs to determine which people can come on board. The first step is to confirm the identity of a passenger to make sure they are who they say they are. Once a passenger's identity has been determined, the second step is verifying any special services the passenger has access to, whether it's flying first-class or visiting the VIP lounge. In the digital world, authentication and authorization accomplish these same goals. Authentication is used to verify that users really are who they represent themselves to be. Once this has been confirmed, authorization is then used to grant the user permission to access different levels of information and perform specific functions, depending on the rules established for different types of users.

2. What is Hashing in Spring Security?

Hashing is the process of generating a string, or hash, from a given message using a mathematical function known as a cryptographic hash function. While there are several hash functions out there, those tailored to hashing passwords need to have four main properties to be secure:

It should be deterministic: the same message processed by the same hash function should always produce the same hash

It's not reversible: it's impractical to generate a message from its hash

It has high entropy: a small change to a message should produce a vastly different hash

And it resists collisions: two different messages should not produce the same hash

A hash function that has all four properties is a strong candidate for password hashing since together they dramatically increase the difficulty in reverse-engineering the password from the hash. Also, though, password hashing functions should be slow. A fast algorithm would aid brute force attacks in which a hacker will attempt to guess a password by hashing and comparing billions (or trillions) of potential passwords per second.

3. What is Salting and why do we use the process of Salting?

Salting hashes sounds like one of the steps of a hash browns recipe, but in cryptography, the expression refers to adding random data to the input of a hash function to guarantee a unique output, the hash, even when the inputs are the same. Consequently, the unique hash produced by adding the salt can protect us against different attack vectors, such as hash table attacks, while slowing down dictionary and brute-force offline attacks. However, there are limitations in the protections that a salt can provide. If the attacker is hitting an online service with a credential stuffing attack, a subset of the brute force attack category, salts won't help at all because the legitimate server is doing the salting+hashing for you. Hashed passwords are not unique to themselves due to the deterministic nature of hash function: when given the same input, the same output is always produced.

4. What is “intercept-url” pattern?

The intercept-url patterns are processed in the order in which they appear in the security configuration models. In the following example any incoming requests that do not match any of the specific patterns would be denied access since "<intercept-url pattern="/**" access="denyAll" />" is the last pattern to be matched. The pattern attribute on the security:intercept-url tag uses Ant paths. Under this syntax has no meaning except for a literal... So, the pattern is most likely not recognized by Spring security and ignored.

5. What do you mean by session management in Spring Security?

The spring security helps us to control the HTTP sessions. Spring security use the following options to control the HTTP session functionalities SessionManagementFilter, SessionAuthneticationStrategy. These 2 helps spring security to manage the following options in the security session:

Session Timeout detection and handling.

Concurrent sessions (how many sessions an authenticated user may have open concurrently).

Session-fixation – handle the session.

6. Why we need Exception Handling?

Java exception handling is important because it helps maintain the normal, desired flow of the program even when unexpected events occur. If Java exceptions are not handled, programs may crash, or requests may fail. This can be very frustrating for customers and if it happens repeatedly, you could lose those customers. The worst situation is if your application crashes while the user is doing any important work, especially if their data is lost. To make the user interface robust, it is important to handle Java exceptions to prevent the application from unexpectedly crashing and losing data. There can be many causes for a sudden crash of the system, such as incorrect or unexpected data input. For example, if we try to add two users with duplicate IDs to the database, we should throw an exception since the action would affect database integrity.

7. Explain what is AuthenticationManager in Spring security?

The AuthenticationManager is the main strategy interface for authentication. If the principal of the input authentication is valid and verified, AuthenticationManager#authenticate returns an Authentication instance with the authenticated flag set to true. Otherwise, if the principal is not valid, it will throw an AuthenticationException. For the last case, it returns null if it can't decide. ProviderManager is the default implementation of AuthenticationManager. It delegates the authentication process to a list of AuthenticationProvider instances. We can set up global or local AuthenticationManager if we extend WebSecurityConfigurerAdapter. For a local AuthenticationManager, we could override configure (AuthenticationManagerBuilder). AuthenticationManagerBuilder is a helper class that eases the setup of UserDetailService, AuthenticationProvider, and other dependencies to build an AuthenticationManager. For a global AuthenticationManager, we should define an AuthenticationManager as a bean.

8. What is Spring Security Filter Chain?

Spring Security's web infrastructure is based entirely on standard servlet filters. It doesn't use servlets or any other servlet-based frameworks (such as Spring MVC) internally, so it has no strong links to any particular web technology. It deals in HttpServletRequests and HttpServletResponses and doesn't care whether the requests come from a browser, a web service client, an HttpInvoker or an AJAX application. Spring Security maintains a filter chain internally where each of the filters has a particular responsibility and filters are added or removed from the configuration depending on which services are required. The ordering of the filters is important as there are dependencies between them. If you have been using namespace configuration, then the filters are automatically configured for you and you don't have to define any Spring beans explicitly but here may be times when you want full control over the security filter chain, either because you are using features which aren't supported in the namespace, or you are using your own customized versions of classes.

9. What are the differences between OAuth2 and JWT?

Developers describe JSON Web Token as "A JSON-based open standard for creating access tokens". JSON Web Token is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. On the other hand, OAuth2 is detailed as "An open standard for access delegation". It is an authorization framework that enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. JSON Web Token and OAuth2 belong to "User Management and Authentication" category of the tech stack.

10. What is method security and why do we need it?

The method level Spring security allows us to add security to individual methods within our service layer. In simple terms, Spring method security allows us to support / add authorization supports at the method level. On a high level, we can configure which roles are allowed to access what method within the same service class. Let’s take an example of CustomerService class. A customer service can only use the view method. We only allow the user with Admin permission to call the delete method in the same service class. Spring security supports both JSR-250 based annotation and Spring security-based annotation, which allows us to use the new and powerful Spring expression language.

11. What Proxy means and how and where can be used?

A proxy server acts as a gateway between you and the internet. It’s an intermediary server separating end users from the websites they browse. Proxy servers provide varying levels of functionality, security, and privacy depending on your use case, needs, or company policy. If you’re using a proxy server, internet traffic flows through the proxy server on its way to the address you requested. The request then comes back through that same proxy server (there are exceptions to this rule), and then the proxy server forwards the data received from the website to you.

12. What is Wrapper Class and where can be used?

Wrapper classes are used to convert any data type into an object. The primitive data types are not objects; they do not belong to any class; they are defined in the language itself. Sometimes, it is required to convert data types into objects in Java language. Wrapper class contains or wraps primitive data type.

13. What is SSL? What is TLS? What is the difference? How can we use them?

SSL stands for “Secure Socket Layer”. Netscape developed the first version of SSL in 1995. SSL is a cryptographic protocol that uses explicit connections to establish secure communication between web server and client. Three versions of SSL have been released: SSL 1.0, 2.0, and 3.0. All versions of SSL have been found vulnerable, and they all have been deprecated.

TLS stands for “Transport Layer Security”. The first version of TLS was developed by the Internet Engineering Taskforce (IETF) in 1999. TLS is also a cryptographic protocol that provides secure communication between web server and client via implicit connections. It’s the successor of SSL protocol. Four versions of TLS have been released: TLS 1.0, 1.1, 1.2, and 1.3. TLS 1.0 and 1.1 have been “broken” and are deprecated as of March 2020. TLS 1.2 is the most widely deployed protocol version.

14. Why do you need the intercept-url?

Most web applications using Spring Security only have a couple of intercept-urls because they only have very basic security requirements. You need to have unauthenticated access to the login and login-error screens and usually some aspect of the public site, so that can be a few URL patterns. Then there's often an admin section. If you need more roles, it's customary to associate them with top level URL path components. Although it's not required, it makes it easier to be sure that resources are appropriately protected.