Homework 5- Mustafa YAŞAR

1-What are Authentication and Authorization?

Authentication

Authentication is the act of validating that users are whom they claim to be. This is the first step in any security process.

Complete an authentication process with:

- Passwords. Usernames and passwords are the most common authentication factors. If a user enters the correct data, the system assumes the identity is valid and grants access.
- One-time pins. Grant access for only one session or transaction.
- Authentication apps. Generate security codes via an outside party that grants access.
- Biometrics. A user presents a fingerprint or eye scan to gain access to the system.

In some instances, systems require the successful verification of more than one factor before granting access. This multi-factor authentication (MFA) requirement is often deployed to increase security beyond what passwords alone can provide.

Authorization

Authorization in system security is the process of giving the user permission to access a specific resource or function. This term is often used interchangeably with access control or client privilege.Giving someone permission to download a particular file on a server or providing individual users with administrative access to an application are good examples of authorization.In secure environments, authorization must always follow authentication. Users should first prove that their identities are genuine before an organization's administrators grant them access to the requested resources.

Authentication	Authorization
Authentication verifies who the user is.	Authorization determines what resources a user can access.
Authentication works through passwords, one-time pins, biometric information, and other information provided or entered by the user.	Authorization works through settings that are implemented and maintained by the organization.
Authentication is the first step of a good identity and access management process.	Authorization always takes place after authentication.
Authentication is visible to and partially changeable by the user.	Authorization isn't visible to or changeable by the user.
Example: By verifying their identity, employees can gain access to an HR application that includes their personal pay information, vacation time, and 401K data.	Example: Once their level of access is authorized, employees and HR managers can access different levels of data based on the permissions set by the organization.

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:

1. It should be deterministic: the same message processed by the same hash function should always produce the same hash
2. It's not reversible: it's impractical to generate a message from its hash
3. It has high entropy: a small change to a message should produce a vastly different hash
4. 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.

Some great hash functions that meet all these criteria are PBKDF2, BCrypt, and SCrypt. But first, let's take a look at some older algorithms and why they are no longer recommended.

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

Salting is a concept that typically pertains to password hashing. Essentially, it's a unique value that can be added to the end of the password to create a different hash value. This adds a layer of security to the hashing process, specifically against brute force attacks. A brute force attack is where a computer or botnet attempt every possible combination of letters and numbers until the password is found.

When salting, the additional value is referred to as a **"salt."**

The idea is that by adding a salt to the end of a password and then hashing it, you've essentially complicated the password cracking process.

Let's look at a quick example.

Say the password I want to salt looks like this:

7X57CKG72JVNSSS9

Your salt is just the word SALT

Before hashing, you add SALT to the end of the data. So, it would look like this:

7X57CKG72JVNSSS9SALT

The hash value is different than it would be for just the plain unsalted password. Remember, even the slightest variation to the data being hashed will result in a different unique hash value. By salting your password you're essentially hiding its real hash value by adding an additional bit of data and altering it.

4- What is "intercept-url" pattern?

The <intercep-url> element defines a pattern which is matched against the URLs of incoming request using am ant path style syntax. The access attribute defines the access requirements for requests matching the given pattern.

Spring Security intercept-url configuration steps.

1. Specify the intercepted URL
2. Specify access rights
3. Specify the access protocol
4. Specify the request method

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

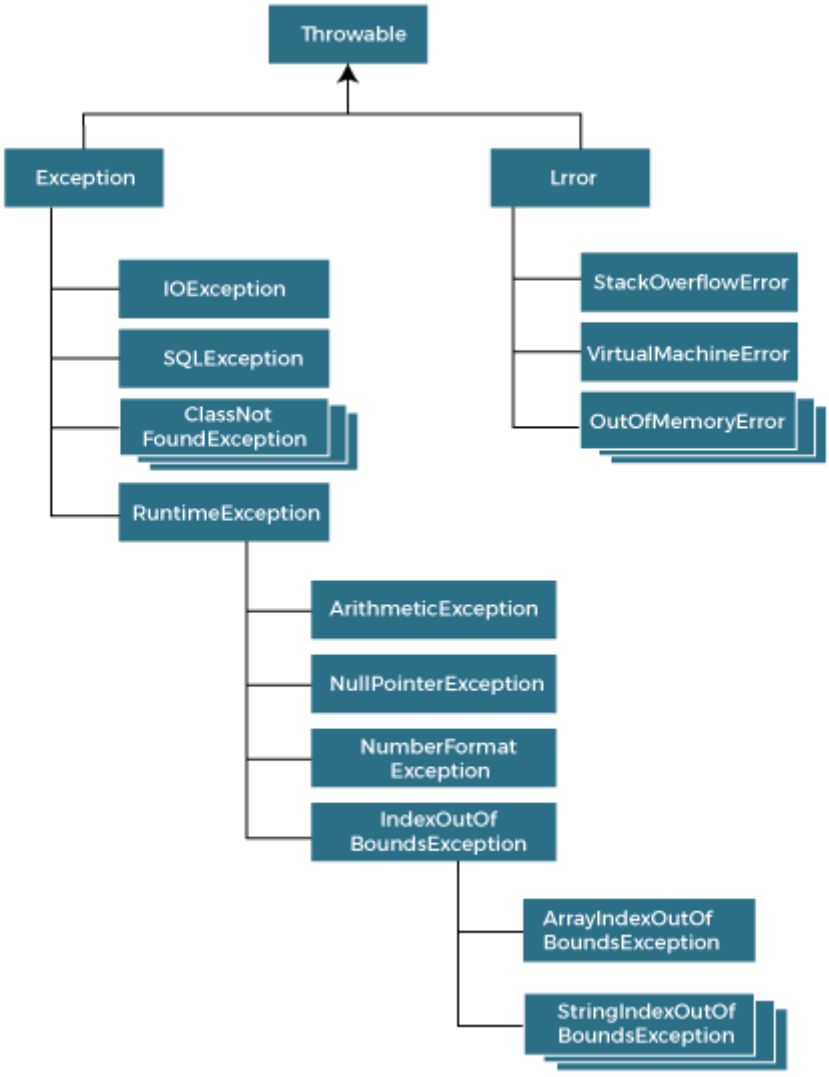
The SessionManagementFilter checks the contents of the SecurityContextRepository against the current contents of the SecurityContextHolder to determine whether a user has been authenticated during the current request, typically by a non-interactive authentication mechanism, such as pre-authentication or remember-me. If the repository contains a security context, the filter does nothing. If it doesn't, and the thread-local SecurityContext contains a (non-anonymous) Authentication object, the filter assumes they have been authenticated by a previous filter in the stack. It will then invoke the configured SessionAuthenticationStrategy.

If the user is not currently authenticated, the filter will check whether an invalid session ID has been requested (because of a timeout, for example) and will redirect to the configured invalidSessionUrl if set.

6- Why we need Exception Handling?

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime. İstisna İşleme, ClassNotFoundException, IOException, SQLException, RemoteException, vb. gibi çalışma zamanı hatalarını işlemek için bir mekanizmadır.

Hierarchy of Java Exception classes



Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

1. Checked Exception
2. Unchecked Exception
3. Error

Difference between Checked and Unchecked Exceptions

1) Checked Exception

The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

2) Unchecked Exception

The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

3) Error

Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

7- Explain what is AuthenticationManager in Spring security?

It is a core interface that spring security uses for the authentication process. It has only one method authenticate which when implemented in a class that implements an Authentication Manager has all the logic for authenticating a user request. The authenticate method takes an “Authentication” object as its parameter and returns an “Authentication” object on successful authentication of the user or else we can have an exception thrown indicating that the user is not authenticated.

8- What is Spring Security Filter Chain?

Spring Security's web infrastructure should only be used by delegating to an instance of FilterChainProxy. The security filters should not be used by themselves In theory you could declare each Spring Security filter bean that you require in your application context file and add a corresponding DelegatingFilterProxy entry to web.xml for each filter, making sure that they are ordered correctly, but this would be cumbersome and would clutter up the web.xml file quickly if you have a lot of filters. FilterChainProxy lets us add a single entry to web.xml and deal entirely with the application context file for managing our web security beans. It is wired using a DelegatingFilterProxy, just like in the example above, but with the filter-name set to the bean name “filterChainProxy”. The filter chain is then declared in the application context with the same bean name.

9- What are the differences between OAuth2 and JWT?

An OAuth token does not always implies an opaque token- a random sequence of alphanumeric characters that contains no inherent meaning. The OAuth token is a security token granted by IDP that can then be validated only by that same OAuth token provider. An opaque token is not the only kind of OAuth token. The opaque token is one kind of token; JWT can be used as another kind of OAuth token that is self-contained.

JWT, in contrast, are not opaque. JWT actually contains meta data that can be extracted and interpreted by any bearer that has the token. JWT usually contains real information so it can be of variable size depending on the claims contained within it and the algorithm used to sign it.

Any holder of the JWT can inspect it, validate it and then optionally make authorization decisions based on the claims presented in it.

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

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.

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

Proxy is a structural pattern that provides a substitute or placeholder for another object in order to control creation and access to the original object. This placeholder object is called proxy object. The proxy pattern allows us to perform some logic either before or after invoking the original object. It is mostly used for postponing the cost of instantiating of an object (especially an expensive to create object) until it is actually needed by clients.

Some types of proxy

Virtual Proxy: Creates expensive objects on demand.

Protection Proxy: Controls access to the real object.

Remote Proxy: Represents an object running on a remote JVM.

Smart reference Proxy: Performs additional actions on the real object, such as maintaining reference counts to a real object so that the real object can be freed when no more references exist.

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

A Wrapper class is a class whose object wraps or contains primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

Need of Wrapper Classes

1. They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).
2. The classes in java.util package handles only objects and hence wrapper classes help in this case also.
3. Data structures in the Collection framework, such as ArrayList and Vector, store only objects (reference types) and not primitive types.
4. An object is needed to support synchronization in multithreading.

Primitive Data types and their Corresponding Wrapper class

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

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

SSL (*Secure Sockets Layer*) is a cryptographic protocol that facilities secure, encrypted connections on the Internet. The most well-known use of SSL is web browsers connecting to websites, where SSL is used on top of HTTP to create a HTTPS connection.

TLS (*Transport Layer Security*) is also a cryptographic protocols that encrypt data and authenticate a connection when transporting data on the Internet. TLS is just a newer and upgraded version of SSL. It fixes some security vulnerabilities in the earlier SSL protocols.

SSL vs TLS

Property	SSL	TLS
Cipher Suites	Offers support for Fortezza cipher suite.	Offers support for RC4, Triple DES, AES, IDEA.
Alert Messages	“No certificate” alert message.	Different alert messages depending on the situation.
Record Protocol	Uses MAC (Message Authentication Code).	Uses HMAC (Hash-Based Message Authentication Code).
Handshake Process	Hash calculation includes the master secret and pad.	Hashes are calculated over the handshake message.
Message Authentication	Ad-hoc method of adjoining key details and application data.	Adjoining key details and application data through HMAC.

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. We 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, and then everything else is ROLE_USER. If we need more roles, we should associate them with top level URL path components.