Spring Security

Spring Security is a framework that provides authentication, authorization, and protection against common attacks.

By Eslam Khder

Spring Security

Authentication and Authorization:

- **Authentication:** Verifies the identity of users (who they are).
- **Authorization:** Controls what authenticated users can access (what they are allowed to do).

In the context of security, a "**role**" typically refers to a defined set of responsibilities, permissions, and access levels assigned to a person or group within an organization or system.

By Eslam Khder

Spring Security

Password Storage and Encoding:

- Provides password encoding mechanisms to store passwords securely.
- Supports password hashing algorithms like BCrypt, SCrypt, and Argon2.

Spring Security

Security Filters

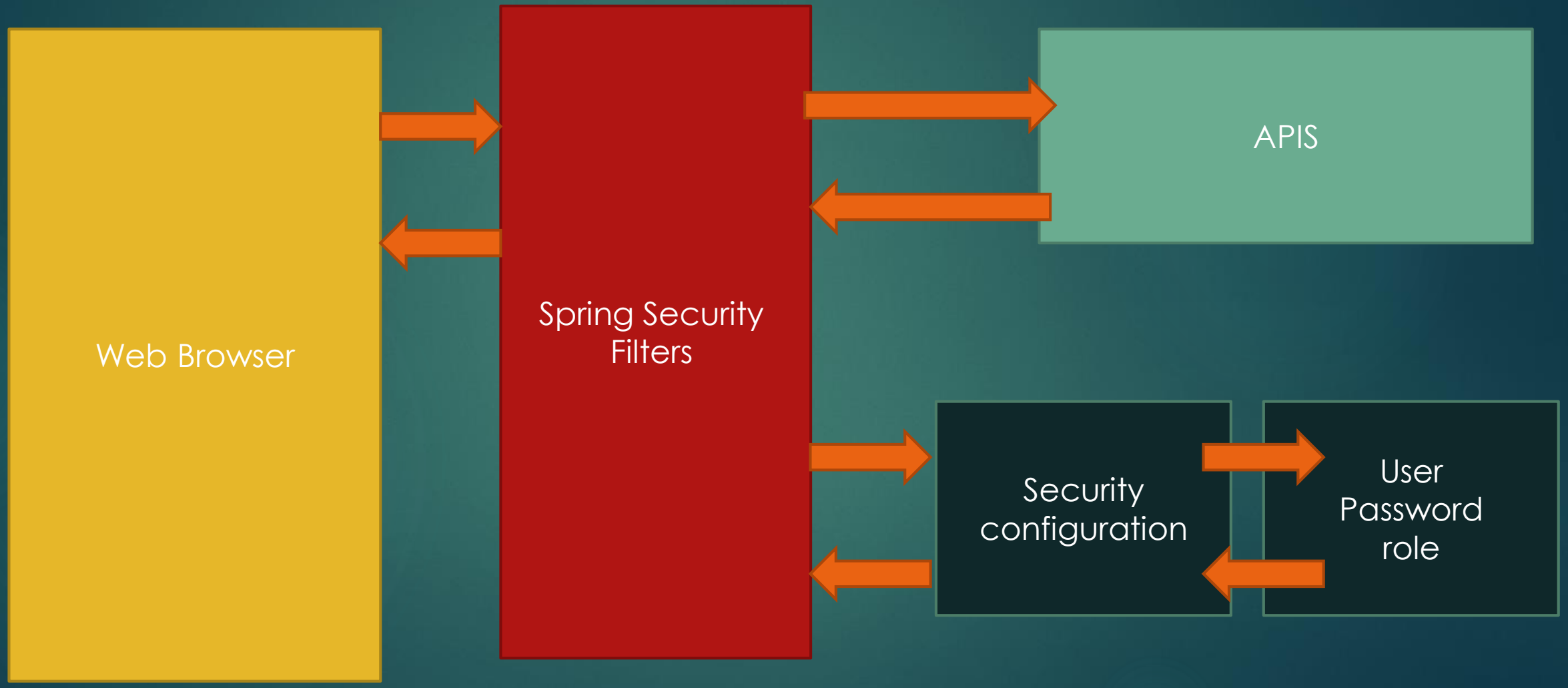
Filters are central to Spring Security's operation. They intercept requests and handle both authentication and authorization.

The **Security Filter Chain** contains various filters like `UsernamePasswordAuthenticationFilter`, `BasicAuthenticationFilter`, and `ExceptionTranslationFilter`.

By Eslam Khder

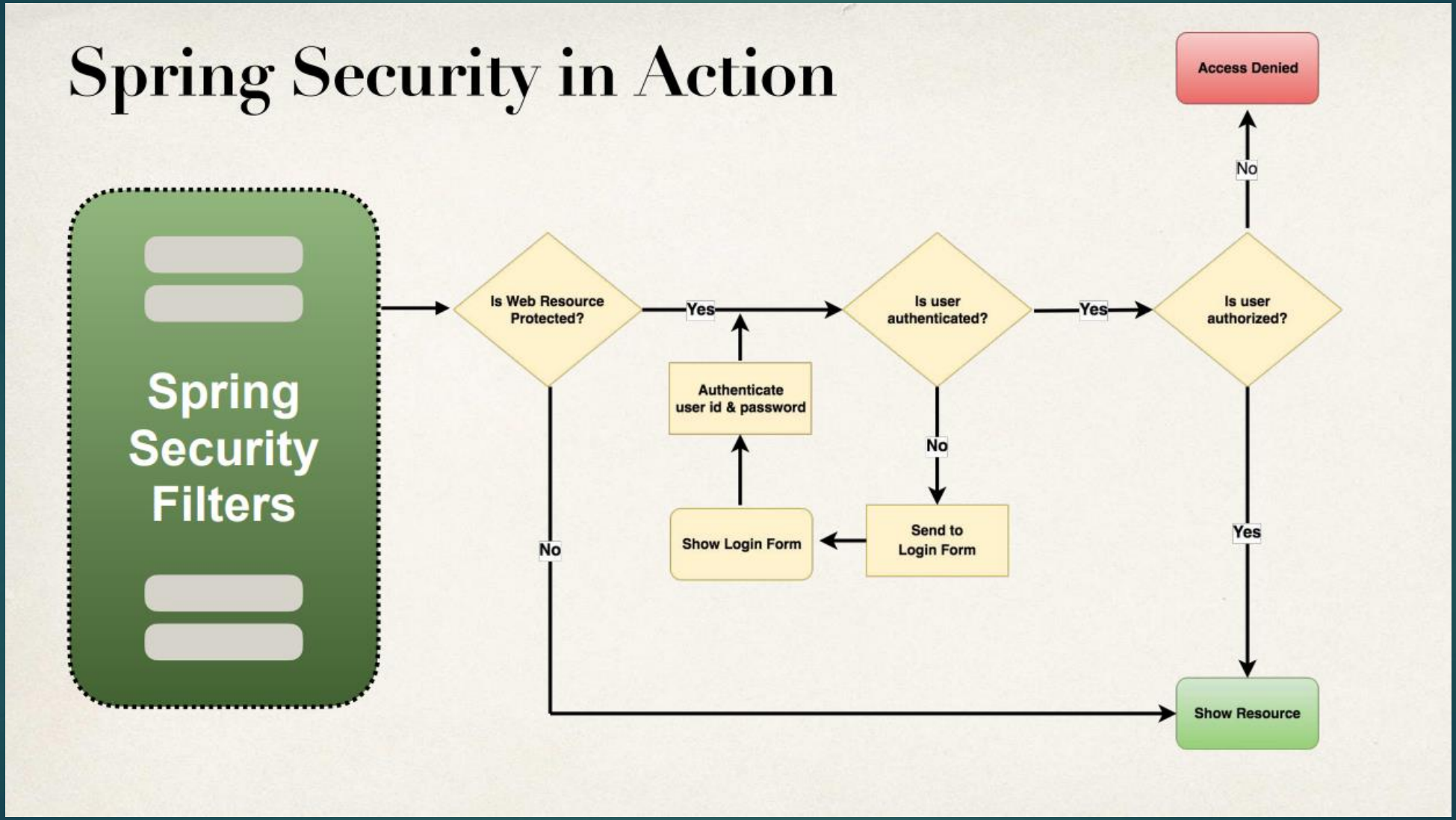
Spring Security defines a framework for security
Implemented using Servlet filters in the background

Spring Security



By Eslam Khder

Spring Security



By Eslam Khder

Spring Security

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

By Eslam Khder

Spring Security

tow type of password

{noop} text plain example :

{bcrypt} bcrypt example :

By Eslam Khder

Spring Security

In Spring Security, `{noop}` and `{bcrypt}` are prefix indicators used in the context of password encoding. They help specify which encoding mechanism was used to encode the password.

Here's an explanation of both, along with examples:

1. `{noop}`: Plain Text Password (No Encoding)

- **What it is**: The `{noop}` prefix indicates that the password is stored in **plain text**, meaning no hashing or encryption has been applied. This is usually used for testing purposes, as storing passwords in plain text is insecure and should not be used in production environments.
- **How to use it**: When Spring Security sees the `{noop}` prefix, it treats the password as a plain-text password.

Example:

plaintext Copy code

```
{noop}password123
```

By Eslam Khder

2. `{bcrypt}`: BCrypt Password Hashing

- **What it is**: The `{bcrypt}` prefix is used to indicate that the password is encoded using the **BCrypt hashing algorithm**. BCrypt is a widely used and secure hashing algorithm designed specifically for password storage, as it is computationally expensive and includes a salt (a random value) to protect against rainbow table attacks.
- **How to use it**: When Spring Security sees the `{bcrypt}` prefix, it uses BCrypt to hash the password. This means the password is not stored as plain text but is hashed using BCrypt, and the hash is stored instead.

Example:

plaintext Copy code

```
{bcrypt}$2a$10$qeS0HEh7urweMojsnwNAR.vcXJeXR1UcMRZ2WcGQl9YeuspUdgF.q
```

if i need it on data base not on memory

Spring Security

users

username	password	enabled

authorities

username	authority

By Eslam Khder

Spring Security

By Eslam Khder

Spring Security

By Eslam Khder

Spring Security

By Eslam Khder

Spring Security

By Eslam Khder

Spring Security

By Eslam Khder

Spring Security

By Eslam Khder

Spring Security

By Eslam Khder

Spring Security

By Eslam Khder

Spring Security

By Eslam Khder