Security

Martin Ledvinka, Petr Křemen, and Lama Saeeda

KBSS

Winter Term 2024

Contents

Spring Security

Spring Security

Spring Security offers the following annotations:

- `@PreFilter` for filtering input iterable argument based on security constraints expressed in SpEL.
- `@PostFilter` for filtering output iterable value based on security constraints expressed in SpEL.
- `@PreAuthorize` for authorizing method execution based on security constraints expressed in SpEL.
- `@PostAuthorize` for authorizing return from the method execution based on security constraints expressed in SpEL.

Spring Security – SpEL

Relevant SpEL expressions:

- `hasRole('ROLE_ADMIN')` checks whether the currently logged in user has the 'ADMIN' role.
- `and`, `or` logical operators.
- `principal` the currently logged in user, e.g., `principal.username`.
- `filterObject` the object filtered from the collection, e.g., `filterObject.customer`.

Become familiar with these annotations before starting the tasks.

Spring Security – Security Filter

Spring Security is based on filters (Sevlet API)

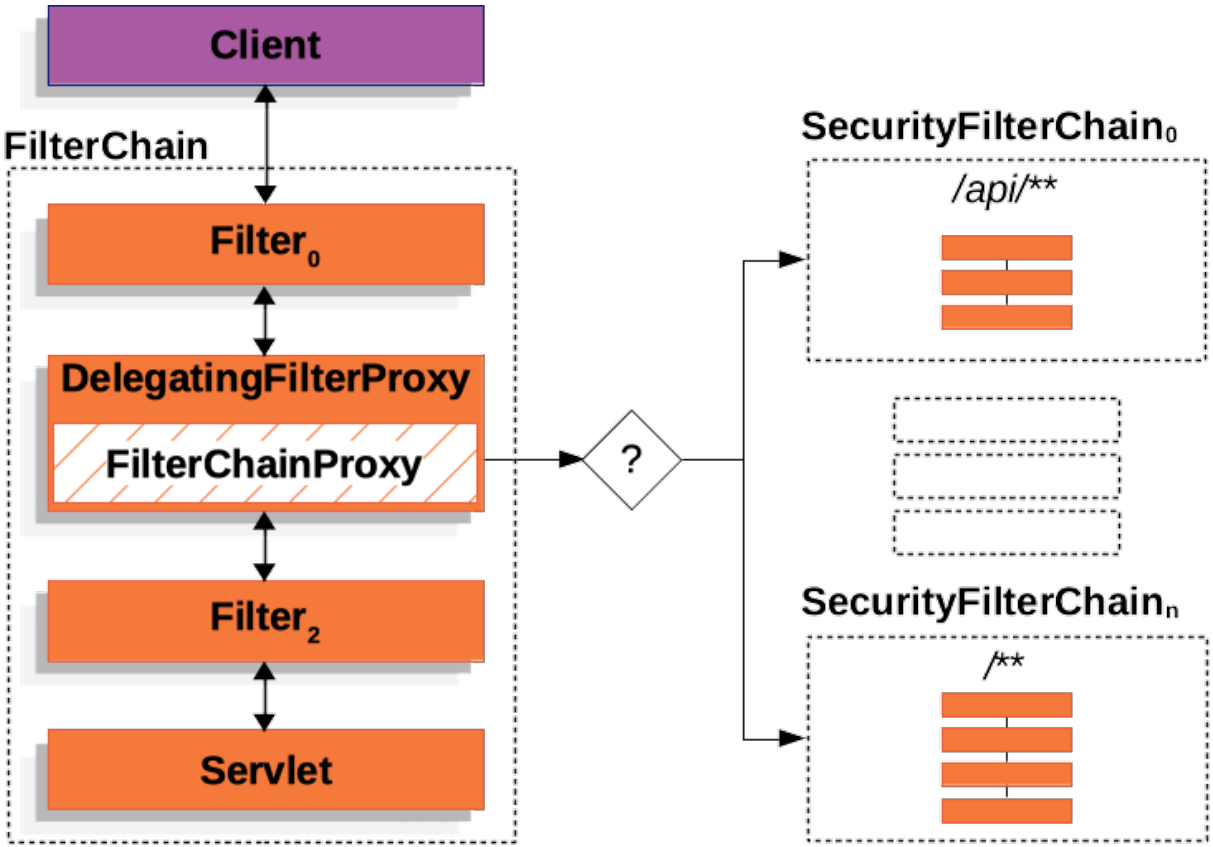

Figure: Source: `https://docs.spring.io/spring-security/reference/servlet/architecture.html`

Spring Security – Authentication Process

See the description at the linked Spring Security docs

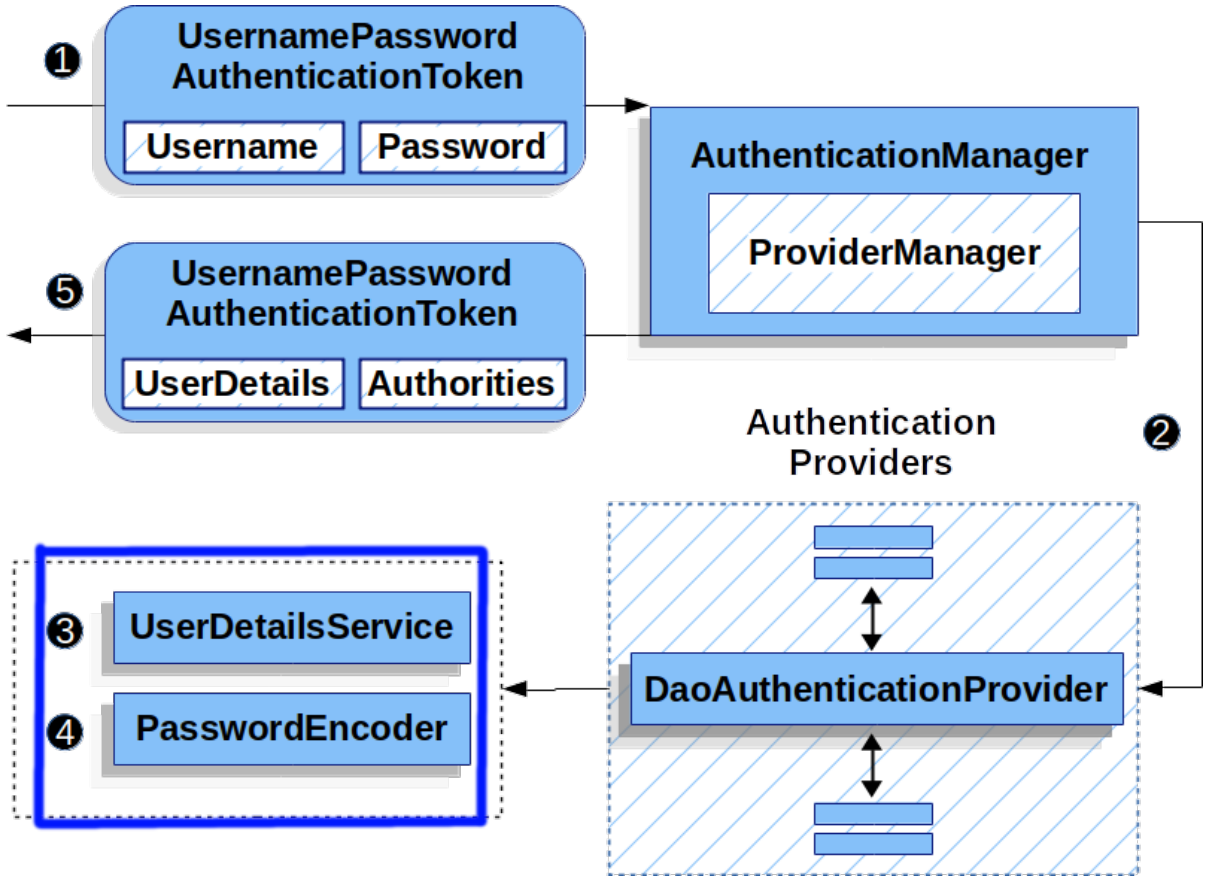

Figure: Source: `https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/dao-authentication-provider.html`

Notes and Possible Pitfalls

- Roles should start with `ROLE_`, e.g., `ROLE_ADMIN`, Spring Security expects this prefix
- CORS needs to be configured if frontend is running on a different host/port than backend
- Security context (i.e., current user) is accessed via static methods and thread-local variables
  - Specific to the thread processing a request

Security in E-shop

- `SecurityConfig` and all the beans used in it
  - Set up security filter (see inline comments in code)
  - Custom success an failure handlers to return JSON (suitable for JS-based frontend)
- `SecurityUtils`
  - Use thread local-based `SecurityContextHolder` to access current security context
- `UserDetailsService`
  - Custom implementation to work with e-shop user account and `UserDetails`
- Aforementioned Spring security annotations

Security in E-shop – Demo

- Let's start e-shop and put some breakpoints to see what happens behind the scene
- Put breakpoints into:
  - `AbstractUserDetailsAuthenticationProvider.authenticate` and log in
  - `AuthorizationFilter.doFilter` and access any API endpoint

Tasks

Syncing Your Fork

1. Ensure you have no uncommitted changes in the working tree
   - `git status` → your branch is up to date, nothing to commit
2. Fetch branches and commits from the upstream repository (EAR/B241-eshop)
   - `git fetch upstream`
3. Check out the task branch from **upstream** (one line!)
   - `git checkout -b b241-seminar-09-task upstream/b241-seminar-09-task`
4. Do the task
5. Commit and push the solution to **your** fork
   - `git push -u origin b241-seminar-09-task`

Task – 1 point

1. Data modifying operations in `CategoryController` should be allowed only for administrators. Use appropriate Spring security features to secure the corresponding endpoints.
2. Method `OrderService.findAll` should return only the orders of the current user or all orders if the current user is administrator.

Acceptance criteria:

1. All tests in `CategoryControllerSecurityTest` pass.
2. All tests in `OrderServiceSecurityTest` pass.

Hints: Administrators are users with `Role.ADMIN`.

Task – 1 point

1. The task branch contains code that is vulnerable to SQL injection
2. Exploit this vulnerability and then fix it
   - You can use e-shop UI or the REST API directly
3. Send an email to your teacher describing the exploit
   - How to exploit the vulnerability?
   - What code did you inject?
   - What was the effect?
4. Fix the vulnerability and push your changes

Hints: Login screen allows accessing the vulnerability

Resources

- https://docs.spring.io/spring-framework/reference/core/expressions.html
- https://docs.spring.io/spring-security/reference/servlet/authorization/method-security.html