Enterprise System Integration
(MTAT.03.229)

LECTURE 11: SECURITY ASPECTS OF INTEGRATION - REST SECURITY (AUTHORISATION AND AUTHENTICATION) - II

MOHAMAD GHRIB
UNIVERSITY OF TARTU
Recap

1. Configure a Spring Security authentication mechanism,
2. Configure a Spring Security authorization mechanism.

Specify App users and associate appropriate roles to them, which define their allowed/prevented activities.
1. Create a simple controller,
2. Configure a Spring Security in memory authentication mechanism: overrides `AuthenticationManagerBuilder` through `inMemoryAuthentication()`.
3. Configure a Spring Security authorization mechanism.

Specify App users and associate appropriate roles to them, which define their allowed/prevented activities.
Spring Boot Security - Authentication through JPA

1. Create a simple controller,
2. Configure a Spring Security in memory authentication mechanism: overrides AuthenticationManagerBuilder through userDetailsService(userDetailsService).
3. Configure a Spring Security authorization mechanism.

Specify App users and associate appropriate roles to them, which define their allowed/prevented activities.
Spring Boot Security - Authentication through JPA

Authentication Manager
  authenticate()

Authentication Provider(s)
  authenticate() supports()

UserDetailsService
  loadUserByUsername()

UserDetails

User

JPA
Spring Boot Security - Authentication through JPA

**<interface> UserDetails**
- getAuthorities()
- getPassword()
- getUsername()
- isAccountNonExpired()
- isAccountNonLocked()
- isCredentialsNonExpired()
- isEnabled()

**WebSecurityConfigurerAdapter**

**<interface> UserDetailsService**
- loadUserByUsername(username)

**MyUserDetailsService**
- repository: userRepository

**SecurityConfiguration**
- userDetailsService()
- passwordEncoder()
- configure(AuthenticationManagerBuilder)
- configure(HttpSecurity)

**User**
- id
- name
- password
- roles

**MyUserDetails**
- user: User

**<interface> UserRepository**
- findByName(username)

**JpaRepository**
OAuth2 is an open standard for authorization.

OAuth2 is a standard that apps can use to provide client applications with "secure delegated access". OAuth2 works over HTTPS and authorizes devices, APIs, servers, and applications with access tokens rather than credentials.

OAuth2 was made famous by HTTP Basic Authentication, where the user is prompted for a username and password.

Does OAuth2 use the same approach?
There are four key actors in OAuth2:

**Resource Owner**: owns the resource in the resource server.

**Resource Server**: stores the resource that an application wants to access.

**Client**: the application that wants to access your the resource.

**Authorization Server**: responsible for the authentication service.
OAuth2

Client sends authorization request to the Authorization Server.

Authorization Server returns a consent dialog saying “do you allow this application to have access to these scopes?” If you already have a cached session cookie, you’ll just see the consent dialog box. View the consent dialog, and agree.

The authorization grant is passed back to the Client, which can be used to obtain the Resource from the RS
1. Create a OAuth2 App (e.g., GitHub, Facebook);
2. Configure your OAuth2 App.
Thank You for your attention

Mohamad Gharib
mohamad.gharib@ut.ee