Spring Cloud Security

Spring Cloud Security

- Spring Boot ve Spring Security **OAuth2** üzerine kuruludur
- Genellikle **merkezi bir kimlik yönetim servisi** ile çalışır
- Birbirleri ile işbirliği yapan **mikroservis bileşenleri için güvenlik kabiliyetleri** sunar

Spring Cloud Security

pom.xml

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

OAuth2 Client

```
@EnableOAuth2Sso
@SpringBootApplication
public class OAuthClientApplication {

...
}
```

Uygulama oauth hizmeti için bir authorization server'a register olmalıdır

```
security:
  oauth2:
    client:
      clientId: bd1c0a783ccdd1c9b9e4
      clientSecret: 1a9030fbca47a5b2c28e92f19050bb77824b5ad1
      accessTokenUri: https://github.com/login/oauth/access_token
      userAuthorizationUri: https://github.com/login/oauth/authorize
      clientAuthenticationScheme: form
    resource:
      userInfoUri: https://api.github.com/user
      preferTokenInfo: false
```

Resource Server

```
@EnableResourceServer
@SpringBootApplication
public class ResourceServerApplication {
...
}
```

```
security:
  oauth2:
    resource:
      userInfoUri: https://api.github.com/user
      preferTokenInfo: false
```

Zuul Proxy Client Token Relay

- Zuul api gateway aynı zamanda **OAuth2** client olarak tanımlanır
- Bu durumda **OAuth2 token**'ları arka taraftaki **downstream servisler**e iletilebilir

```
@EnableOAuth2Sso
@EnableZuulProxy
@SpringBootApplication
public class OAuthClientApplication {

...
}
```

Kimliklendirilmiş kullanıcıdan elde edilen access token request header'a set edilerek downstream servis'lere iletilir

Resource Server Token Relay

- **service-to-service** çağrıları arasında token relay işlemi söz konusudur
- Uzaktaki servisi çağırmak için restTemplate'in **doğru context** ile oluşturulması yeterlidir

Eğer servis gelen token'ları doğrulamak için UserInfoTokenServices kullanıyorsa bu yöntem kullanılabilir

```
@Bean
public OAuth2RestTemplate restTemplate(
                                UserInfoRestTemplateFactory factory) {
    return factory.getUserInfoRestTemplate();
}
```

Resource Server Token Relay

Eğer servis token doğrulama için UserInfoTokenServices kullanmıyor ise token'ı authorization header değeri olarak set eden bir interceptor yazılmalı ve RestTemplate bu interceptor ile yaratılmalıdır

```
@Bean
public RestTemplate restTemplate(RestTemplateBuilder restTemplateBuilder) {
     RestTemplate restTemplate = restTemplateBuilder.build();
     restTemplate.setInterceptors(Arrays.asList(new
                                        Oauth2AccessTokenRelayRequestInterceptor()));
     return restTemplate;
}
```

Resource Server Token Relay

```
public class Oauth2AccessTokenRelayRequestInterceptor
                                        implements ClientHttpRequestInterceptor {

    @Override
    public ClientHttpResponse intercept(
            HttpRequest request, byte[] body, ClientHttpRequestExecution execution)
                                                        throws IOException {

        Authentication auth = SecurityContextHolder.getContext()
                              .getAuthentication();
        if (auth != null) {
            if (auth.getDetails() instanceof OAuth2AuthenticationDetails) {
                OAuth2AuthenticationDetails details =
                        (OAuth2AuthenticationDetails)auth.getDetails();
                String headerValue =
                        details.getTokenType() + " " + details.getTokenValue();
            request.getHeaders().add(HttpHeaders.AUTHORIZATION,headerValue);
            }
        }
        return execution.execute(request, body);
    }
}
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com