JWT 토큰 기반 인증인가 w/ Keycloak Authz-svr

마이크로서비스와 OAuth2 구성요소인 Authorization Server/ Client/ Resource Server 를 활용해 Single Sign-On 구현 모형을 실습한다. 단일 접점인 Gateway 가 Client 가 되고, 각 마이크로서비스가 Resource Server 에 해당된다. 그리고 Authorization Server 로는 Keycloak 을 활용한다. 본 랩에서는 Gateway 가 Client 와 Resource Server 역할을 가진다.

Instruction

JWT 토큰기반 인증 w/ Keycloak

OAuth2 Stackholders

- **Spring Security 와 Spring oauth2 를 사용하고, Resource Owner, Client, Authorization Server, Resource Server 간의 인증/인가를 실습한다.**
- **여기서 Resouce 란 Gateway 를 경유하는 Rest APIs 를 말한다.**
- **JWT 기반 Access_Token 을 활용한다.**
- **이번 랩에서는 Gateway 를 Client 와 Resource Server 역할로 설정한다.**
- **인증/인가 서버로는 Keycloak(https://www.keycloak.org/) 서버를 활용한다.**

OAuth2 인증/인가(Keycloak) Endpoint 설정

- **Gateway 서비스의 application.yml 파일을 열어본다.**
- **인증/인가를 위한 Authorization Sever 의 Endpoint 가 등록된다.**

```
security:
  oauth2:
    client:
      provider:
        my-keycloak-provider:
          issuer-uri: http://localhost:8080/realms/my_realm
```

- **KeyCloak 에 등록된 Client(Gateway)의 Credential 정보가 설정된다.**
- **OAuth2 의 Grant Type 을 password 방식으로 설정한다.**

```
        keycloak-spring-gateway-client:
          provider: my-keycloak-provider
          client-id: my_client
          client-secret: HKFKYP7kb8OMldAgfvnk27FhRPOv8Y7H
          authorization-grant-type: password
```

OAuth2 Security 상세설정

- Gateway 서비스의 SecurityConfig.java 파일을 열어본다.
- spring-cloud-gateway 는 webflux 로 기동되기 때문에 @EnableWebFluxSecurity 를 적용한다.
- ServerHttpSecurity 생성시, 접근제어목록(ACL)을 적용한다.
- .oauth2Login() OAuth2 의 디폴트 로그인 설정이 적용된다.
- .oauth2ResourceServer() 리소스서버 역할을 부여하고 jwt 형식의 Authorization 을 지정한다.

서비스 구동

- 먼저 Keycloak 서버를 구동한다.

```
cd keycloak/bin
./kc.sh start-dev
```

- keycloak 서버의 default 포트인 8080 으로 실행된다.
- 포트 확인 (Labs > 포트확인)
- Gateway, Order 서비스를 구동한다.

```
cd gateway
mvn spring-boot:run
cd order
mvn spring-boot:run
```

- 각각 8088, 8081 포트로 기동된다.

Protected 리소스 접근

- Security ACL 설정(SecurityConfig.java)에 따라 Gateway 서버나 주문서비스에 접근해 본다.

```
http http://localhost:8088
http http://localhost:8088/orders
```

- 401(Unauthorized) 접근오류 응답이 내려온다.
- 허가된 리소스에 접근해 본다.

```
http http://localhost:8088/test/permitAll
```

- 접근 가능하다.

JWT access_token 발급

- Keycloak 의 인증/인가 Endpoint 에 토큰을 요청한다.

- OAuth2 의 Grant type 은 'password'로 요청한다.
- Keycloak 에 기 등록된 Client 정보와 사용자 정보를 제공한다.

```
curl -X POST "http://localhost:8080/realms/my_realm/protocol/openid-
connect/token" \
--header "Content-Type: application/x-www-form-urlencoded" \
--data-urlencode "grant_type=password" \
--data-urlencode "client_id=my_client" \
--data-urlencode "client_secret=HKFKYP7kb8OMldAgfvnk27FhRPOv8Y7H" \
--data-urlencode "username=user@uengine.org" \
--data-urlencode "password=1"
```

- 응답으로 access_token 과 refresh_token 이 내려온다.
- 출력된 access_token 을 복사하여 https://jwt.io/ 페이지에 접속 후 decode 해 본다.

Header, Payload, Signature 로 파싱된다.

- user@uengine.org 계정이 가진 Role 은 ROLE_USER 임을 확인한다.

access_token 으로 Protected 리소스 접근

- access_token 을 복사하여 Request Header 에 넣어 Protected 리소스에 접근한다.

```
export access_token=[ACCESS_TOKEN]
echo $access_token
http localhost:8088/orders "Authorization: Bearer $access_token"
http localhost:8088/test/user "Authorization: Bearer $access_token"
http localhost:8088/test/authenticated "Authorization: Bearer
$access_token"
http localhost:8088/test/admin "Authorization: Bearer $access_token"
```

- '/test/admin' 리소스는 권한이 불충분(403 Fobidden)하여 접근할 수 없다.
- 관리자 권한이 있는 계정으로 다시 한번 토큰을 요청한다.

```
curl -X POST "http://localhost:8080/realms/my_realm/protocol/openid-
connect/token" \
--header "Content-Type: application/x-www-form-urlencoded" \
--data-urlencode "grant_type=password" \
--data-urlencode "client_id=my_client" \
--data-urlencode "client_secret=HKFKYP7kb8OMldAgfvnk27FhRPOv8Y7H" \
--data-urlencode "username=admin@uengine.org" \
--data-urlencode "password=1"
```

- access_token 을 복사하여 Request Header 에 넣어 Protected 리소스에 접근한다.

```
export access_token=[ACCESS_TOKEN]
http localhost:8088/test/admin "Authorization: Bearer $access_token"
```

- 정상적으로 접근이 가능하다.

Wrap up

- Gateway 가 리소스서버 역할까지 수행하므로 각 마이크로서비스 리소스들의 Fine grained 한 접근제어를 Gateway 에서 관리

- 이로 인해 ACL 정보 가독성이 떨어지거나, ACL 오류발생 시 잠재적 분쟁 소지
- MSA 별 Autonomous ACL 관리책임 분산을 위해 인증 및 인가를 분리하는 정책 권고
- Gateway 는 인증을 포함한 Coarse grained ACL Policy 를 담당하고, 각 MSA 에서 Fine grained 한 ACL Policy 적용

Service Clear

- 다음 Lab 을 위해 기동된 모든 서비스 종료

```
fuser -k 8080/tcp
fuser -k 8081/tcp
fuser -k 8088/tcp
```