Piotr's TechBlog

Microservices security with Oauth2

Preface

One of the most important aspects to consider when exposing a public access API consisting of many microservices is security. Spring has some interesting features and frameworks which makes configuration of our microservices security easier. In this article I'm going to show you how to use Spring Cloud and Oauth2 to provide token access security behind API gateway.

Theory

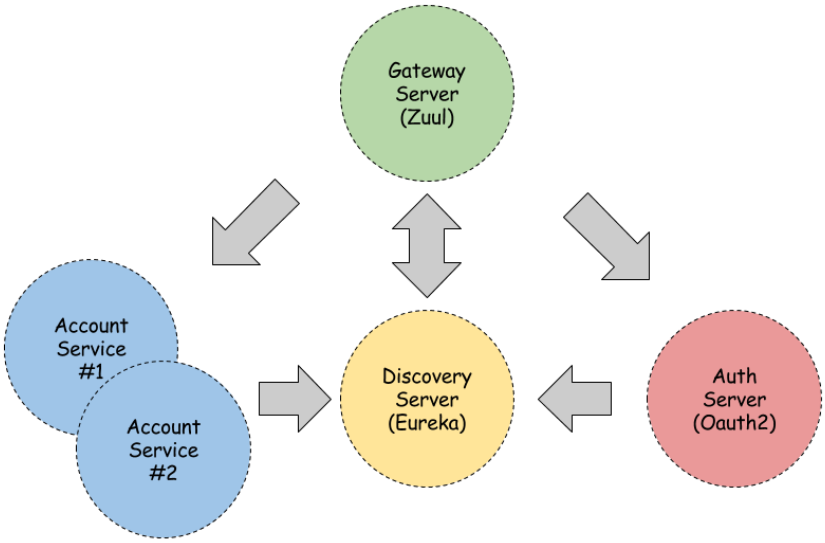
OAuth2 standard is currently used by all the major websites that allow you to access their resources through the shared API. It is an open authorization standard allowing users to share their private resources stored in one page to another page without having to go into the service of their credentials. These are basic terms related to oauth2.

- **Resource Owner** – dispose of access to the resource
- **Resource Server** – server that stores the owner's resources that can be shared using special token
- **Authorization Server** – manages the allocation of keys, tokens and other temporary resource access codes. It also has to ensure that access is granted to the relevant person
- **Access Token** – the key that allows access to a resource
- **Authorization Grant** – grants permission for access. There are different ways to confirm access: authorization code, implicit, resource owner password credentials, and client credentials

You can read more about this standard here and in this digitalocean article. The flow of this protocol has three main steps. In the begining we authorization request is sent to Resource Owner. After response from Resource Owner we send authorization grant request to Authorization Server and receive access token. Finally, we send this access token to Resource Server and if it is valid the API serves the resource to the application.

Our solution

The picture below shows architecture of our sample. We have API Gateway (Zuul) which proxies our requests to authorization server and two instances of account microservice. Authorization server is some kind of infrastructure service which provides outh2 security mechanisms. We also have discovery service (Eureka) where all of our microservices are registered.



Gateway

For our sample we won't provide any security on API gateway. It just has to proxy requests from clients to authorization server and account microservices. In the Zuul's gateway configuration visible below we set *sensitiveHeaders* property on

empty value to enable *Authorization* HTTP header forward. By default Zuul cut that header while forwarding our request to the target API which is incorrect because of the basic authorization demanded by our services behind gateway.

```
zuul:
  routes:
    uaa:
      path: /uaa/**
      sensitiveHeaders:
      serviceId: auth-server
    account:
      path: /account/**
      sensitiveHeaders:
      serviceId: account-service
```

Main class inside gateway source code is very simple. It only has to enable Zuul proxy feature and discovery client for collecting services from Eureka registry.

```
@SpringBootApplication
@EnableZuulProxy
@EnableDiscoveryClient
public class GatewayServer {

    public static void main(String[] args) {
        SpringApplication.run(GatewayServer.class, args);
    }

}
```

Authorization Server

Our authorization server is as simple as possible. It based on default Spring security configuration. Client authorization details are stored in an in-memory repository. Of cource in the production mode you would like to use other implementations instead of in-memory repository like JDBC datasource and token store. You can read more about Spring authorization mechanisms in Spring Security Reference and Spring Boot Security. Here's fragment of configuration from *application.yml*. We provided user basic authentication data and basic security credentials for the */token* endpoint: `client-id` and `client-secret`. The user credentials are the normal Spring Security user details.

```
security:
  user:
```

```
    name: root
    password: password
  oauth2:
    client:
      client-id: acme
      client-secret: secret
```

Here's main class of our authentication server with `@EnableAuthorizationServer`. We also exposed one REST endpoint with user authentication details for account service and enabled Eureka registration and discovery for clients.

```
@SpringBootApplication
@EnableAuthorizationServer
@EnableDiscoveryClient
@EnableResourceServer
@RestController
public class AuthServer {

    public static void main(String[] args) {
        SpringApplication.run(AuthServer.class, args);
    }

    @RequestMapping("/user")
    public Principal user(Principal user) {
        return user;
    }

}
```

Application – account microservice

Our sample microservice has only one endpoint for @GET request which always returns the same account. In main class resource server and Eureka discovery are enabled. Service configuration is trivial. Sample application source code is available on GitHub.

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableResourceServer
public class AccountService {

    public static void main(String[] args) {
        SpringApplication.run(AccountService.class, args);
    }

}
```

```
security:
```

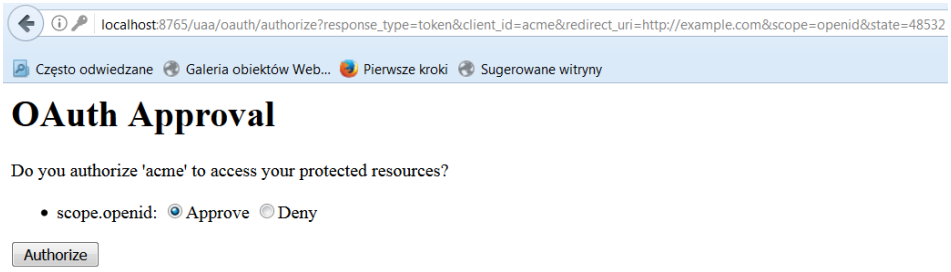
```
  user:
    name: root
    password: password
  oauth2:
    resource:
      loadBalanced: true
      userInfoUri: http://localhost:9999/user
```

Testing

We only need web browser and REST client (for example Chrome Advanced REST client) to test our solution. Let's start from sending authorization request to resource owner. We can call oauth2 authorize endpoint via Zuul gateway in the web browser.

http://localhost:8765/uaa/oauth/authorize?response_type=token&client_id=acme&redirect_uri=http://example.com&scope=openid&state=48532

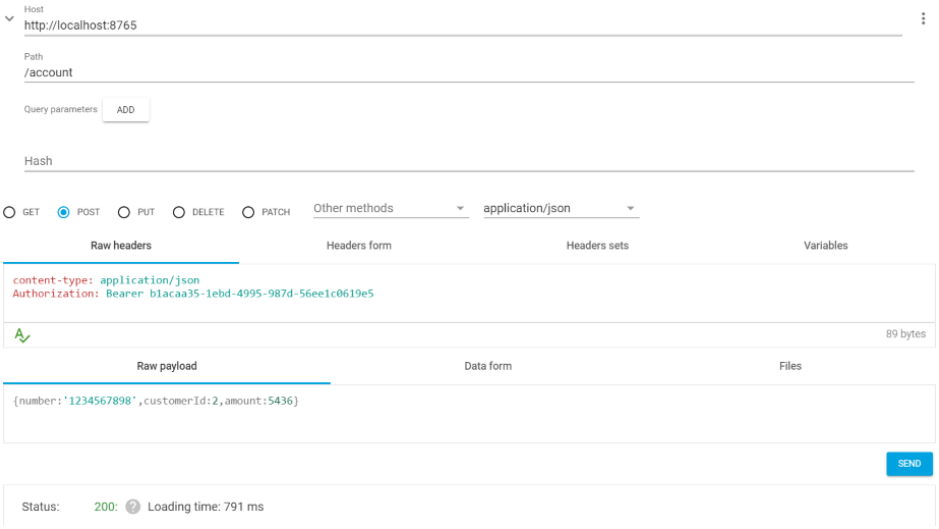
After sending this request we should see page below. Select *Approve* and click Authorize for requests an access token from the authorization server. If the application identity is authenticated and the authorization grant is valid an access token to the application should be returned in the HTTP response.



*http://example.com/#access_token=**b1acaa35-1ebd-4995-987d-56ee1c0619e5**&token_type=bearer&state=48532&expires_in=43199*

And the final step is to call account endpoint using access token. We had to put it into Authorization header as bearer token. In the sample application logging level

for security operation is set to TRACE so you can easily find out what happened if something goes wrong.



Conclusion

To be honest I'm not very familiar with security issues in applications. So one very important thing for me is the simplicity of security solution I decided to use. In Spring Security we have almost all needed mechanisms out of the box. It also provides components which can be easily extendable for more advanced requirements. You should treat this article as a brief introduction to more advanced solutions using Spring Cloud and Spring Security projects.

Share this:

Related

Part 2: Microservices security with OAuth2

In "security"

Building Secure APIs with Vert.x and OAuth2

In "security"

Advanced Microservices Security with OAuth2

In "microservices"

Author: Piotr Mińkowski

IT Architect, Java Software Developer View all posts by Piotr Mińkowski

 Piotr Mińkowski / February 22, 2017 / security / java, Oauth2, security, spring, spring cloud, spring-security, Zuul

34 thoughts on “Microservices security with Oauth2”

 Dhiraj Ray

May 5, 2017 at 8:30 am

There are many articles out there on this topic but this article explains it in a best way.

★ Like

 **Sayali Shinde**

September 9, 2017 at 1:17 pm

What software do we need to install ? Do we need to install Eureka or Zuul ?

★ Like

 **Piotr Mińkowski**

September 9, 2017 at 9:26 pm

Well, you don’t have to install it. You just run services with them

★ Like

 **Jaime**

November 2, 2017 at 4:15 pm

I get the token after login in browser, OK.
But when i put the token in postman with "Bearer token_string", it returns the login form, why?
Another question, after i login in the browser i cant access via zuul to my microservices, it says Full authentication is required to access this resource, please give me an idea of what is going wrong.

Greetings from Perú

★ Like

Jaime

November 3, 2017 at 3:58 am

forget it, i did it 🙂

★ Like

Piotr Mińkowski

November 3, 2017 at 1:23 pm

Ok 🙂

★ Like

Vetri

January 16, 2018 at 6:41 am

can you help me.. got same problem/..

★ Like

Piotr Mińkowski

January 18, 2018 at 10:54 pm

If you send header Authorization: Bearer it should works...

★ Like

Vetri

January 16, 2018 at 6:51 am

Invalid CSRF Token 'null' was found on the request parameter '_csrf' or header 'X-CSRF-TOKEN

★ Like

Rake

November 19, 2017 at 10:05 am

I get the token after login in browser, OK.
But when i put the token in postman with "Bearer token_string", it returns the login form, why?

i am gettng same error, can you tell me what wrong i am doing

★ Like

Rake

November 20, 2017 at 7:24 am

Hi Can you please update whats wrong i am doing. Its not working from application also. Getting login page in response while calling account endpoint using access token.

★ Like

Rake

November 20, 2017 at 12:40 pm

I am calling following endpoint on account-service . I am using Rest Client chrome extension to call endpoint.
http://localhost:8765/account
Request Type: Get
Header passed:
Authorization:Bearer 20cbd6ae-7b5f-47dc-860f-c047e36ab3a6
Content-Type:application/json
Response getting:
==========
Login page
Login page
Example user: user / password
Username:
Password:
Back to home page
==========

★ Like

Piotr Mińkowski

November 20, 2017 at 11:02 am

Can you paste the request which is sent to the service (with HTTP headers)?

★ Like

Rake

November 20, 2017 at 11:33 am

I am calling following endpoint on account-service . I am using Rest Client chrome extension to call endpoint.
http://localhost:8765/account

Request Type: Get
Header passed:
Authorization:Bearer 20cbd6ae-7b5f-47dc-860f-c047e36ab3a6
Content-Type:application/json

Response getting:

Login page

Login page
Example user: user / password

Username:

Password:

<!– –>

Back to home page

★ Like

Rake

November 20, 2017 at 7:25 am

Hi Can you please update whats wrong i am doing. Its not working from application also. Getting login page in response while calling account endpoint using access token.

★ Like

Rake

November 20, 2017 at 11:45 am

i am calling following to get token

http://localhost:8765/uaa/oauth/authorize?response_type=token&client_id=acme&redirect_uri=http://example.com&scope=openid&state=48532

★ Like

Piotr Mińkowski

November 20, 2017 at 9:06 pm

Any logs from auth server?

★ Like

Abul Fayes

January 16, 2018 at 2:23 pm

Where did my comment go?

★ Like

Abul Fayes

January 16, 2018 at 2:24 pm

Getting a JDBC error in the AuthServer:
Caused by: java.net.ConnectException: Operation timed out (Connection timed out)
at java.net.PlainSocketImpl.socketConnect(Native Method) ~[na:1.8.0_121]
at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350) ~[na:1.8.0_121]
at
java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206) ~[na:1.8.0_121]
at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188) ~[na:1.8.0_121]
at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392) ~[na:1.8.0_121]
at java.net.Socket.connect(Socket.java:589) ~[na:1.8.0_121]
at com.mysql.jdbc.StandardSocketFactory.connect(StandardSocketFactory.java:211) ~[mysql-connector-java-5.1.40.jar:5.1.40]
at com.mysql.jdbc.MysqlIO.(MysqlIO.java:300) ~[mysql-connector-java-5.1.40.jar:5.1.40]
... 39 common frames omitted

★ Like

Piotr Mińkowski

January 18, 2018 at 10:55 pm

Have you got running MySQL database?

★ Like

Abul Fayes

January 16, 2018 at 2:25 pm

Would be good to have a docker-compose file for easily starting up the services

★ Like

Piotr Mińkowski

January 18, 2018 at 10:55 pm

You're right 🙂

★ Like

Pingback: Part 2: Microservices security with OAuth2 – paylitrefattand

Alphan ARSLAN

January 25, 2018 at 12:03 pm

It is really great article thanks or sharing

★ Like

Pedro

February 12, 2018 at 2:42 pm

Great Article!

Could you please explain me why we have to define the authorization server as a resource (@EnableResourceServer annotation)? I have test it and works like a charm but I don't understand why this annotation is needed. I understand its use in the microservices but not in the authorization server.

Thanks!

★ Like

Piotr Mińkowski

March 2, 2018 at 8:44 pm

Hi,
Thanks. You don't have to. You can devide seprate out resource server from authorization server

★ Like

Daxol

March 22, 2018 at 9:10 pm

Jak autoryzować się tym tokenem w innych mikroserwisach?

★ Like

Piotr Mińkowski

March 24, 2018 at 10:14 am

Take a look on that article:
https://piotrminkowski.wordpress.com/2017/12/01/part-2-microservices-security-with-oauth2/

★ Like

HB

April 24, 2018 at 11:29 am

hi,
firstly I start by thanking you for the quality of your posts.

I am interested in this post:
https://piotrminkowski.wordpress.com/2017/12/01/part-2-microservices-security-with-oauth2/
but I can not find the source code ?
to make a comparison with the logic of the post
can you help me with that?

thank you in advance.

★ Like

Piotr Mińkowski

April 24, 2018 at 10:22 pm

Hi,
Source code is available on GitHub here: https://github.com/piomin/sample-spring-oauth2-microservices.git. You have three branches with different examples (basic and more advanced). The newest one is with_database. You can also find some more examples related to oauth2 in my blog: https://piotrminkowski.wordpress.com/?s=oauth2

★ Like

HB

April 25, 2018 at 9:51 am

Perfect 🙂

Thanks.

★ Like

DEEPAK

August 31, 2018 at 11:09 am

HI
Thanks for the tutorial.
I have 2 queries .
1. You have commented the code in OAuth2Config.java file in auth package. Any reasons. Will it work .
2. OAUTH2 is authorization framework. So if a website gives option for login with facebook account or google account like delegated the authentication to these sites then will it be still OAuth2 scenario.
Thanks

★ Like

nick ao

October 25, 2018 at 1:51 pm

Hello Piotr Mińkowski,
First of all, i want to thanks you about your post. It's great.
Could you explain me this point, please? I don'n understand why do you put login form and login confirmation in Gateway module. In fact, gateway only have one mission – forwarding the request. And the Auth-Service normaly is the responsible of Authentification and Authorization. So, i think it's better that Auth-Service do Authentification and Authorization and that Gateway only forward the request (it shouldn't have the login form and login confirm).
Thanks a lot
I'm waiting your response.
Best,

★ Like

Piotr Mińkowski

October 30, 2018 at 10:45 pm

Hello. Why do you think that gateway cannot perform authorization or authentication?

★ Like