

Getting started **with** oAuth **2.0**

Tomasz Miler, SAP
29 Październik, 2019

PUBLIC



Agenda



Hack Your Career

**Fit yourself in IT –
Rozegraj swoją
rekrutację**

15.10.2019

Marek Nawa
Maciej Ogrodnik

**Machine Learning od
podstaw**

20.11.2019

Michał Lipka

**Getting started with
OAUTH 2.0**

29.10.2019

Tomasz Miler

**Building a city with
SCRUM - Workshop**

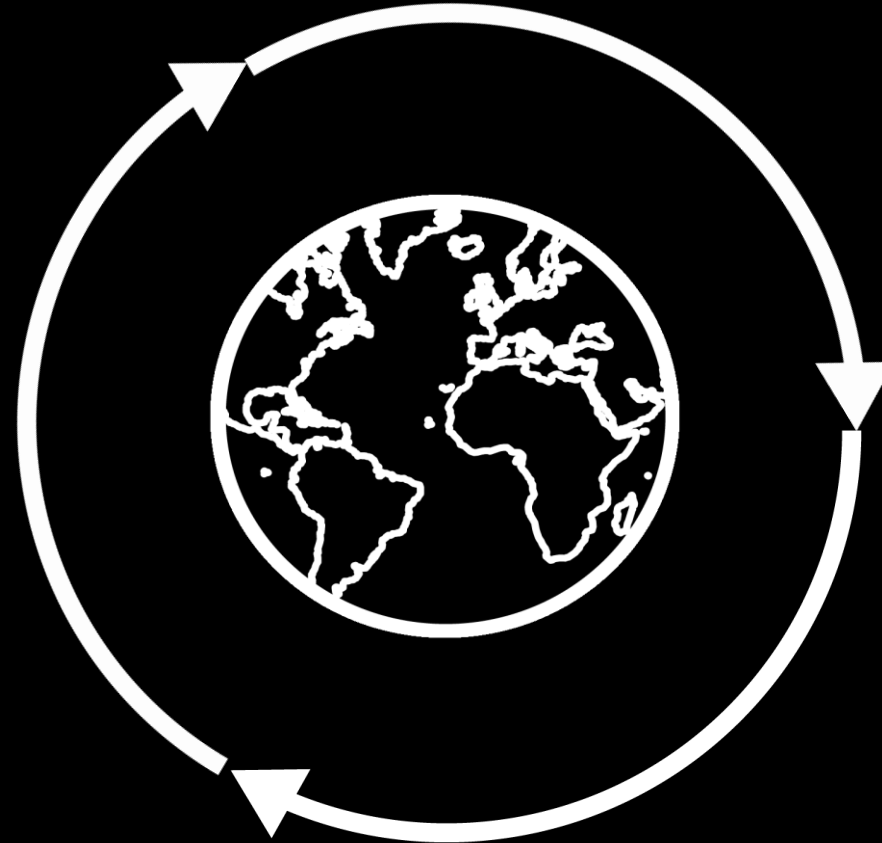
12.12.2019

Michał Drzewiecki

SAP Labs Poland

Top ecommerce,
marketing, billing

Development: Go, Java,
Cloud Native solutions



> 400 pracowników

Najlepszy Pracodawca
w rankingu AON

Jedno z 20 centrów
SAP's Labs Network

Inspiracja



Getting Started with OAuth 2.0

by Scott Brady

Kim jestem?



Tomasz Miler
Senior Software Engineer

Agenda

Problem autoryzacji API

Poprzednie rozwiązania

Autentykacja vs Autoryzacja

OAuth 2.0

Obsługa błędów

Co dalej?

Kiedyś było inaczej

Kiedyś

XML

SAOP

SAML

Dziś

JSON

REST

oAuth 2.0 & Open ID connect

Słownictwo



Authentication vs Authorization
Uwierzytelnienie vs Upoważnienie

Do czego można wykorzystać API?

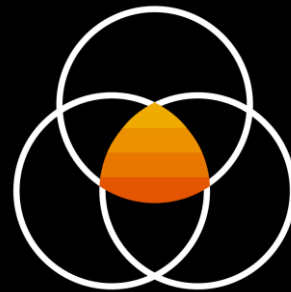
Udostępnianie loginu i hasła

Udostępnianie loginu i hasła

Problemy?



Personifikacja



Jednolite konto



Przechowywanie haseł

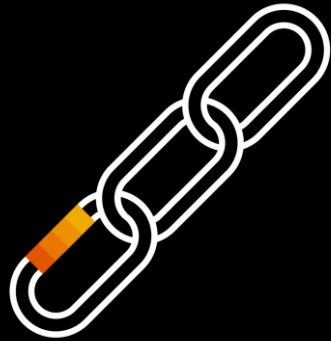


Ciasteczka

Problemy?

Cross-Site Request Forgery (CSRF)

API KEYS



Kompatybilność



Konkretne uprawnienia

API KEYS

Problemy?



Brak standardu

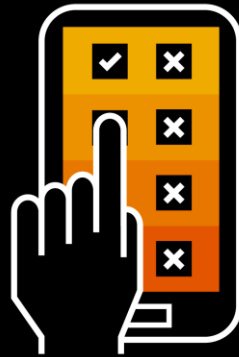


Czas wygaśnięcia

OAUTH 2.0



Ustandaryzowany



Zarządzanie uprawnieniami



Stworzony do API



Delegacja uprawnień

OAUTH 2.0



Aplikacja kliencka
Client application



Właściciel danych
Resource owner

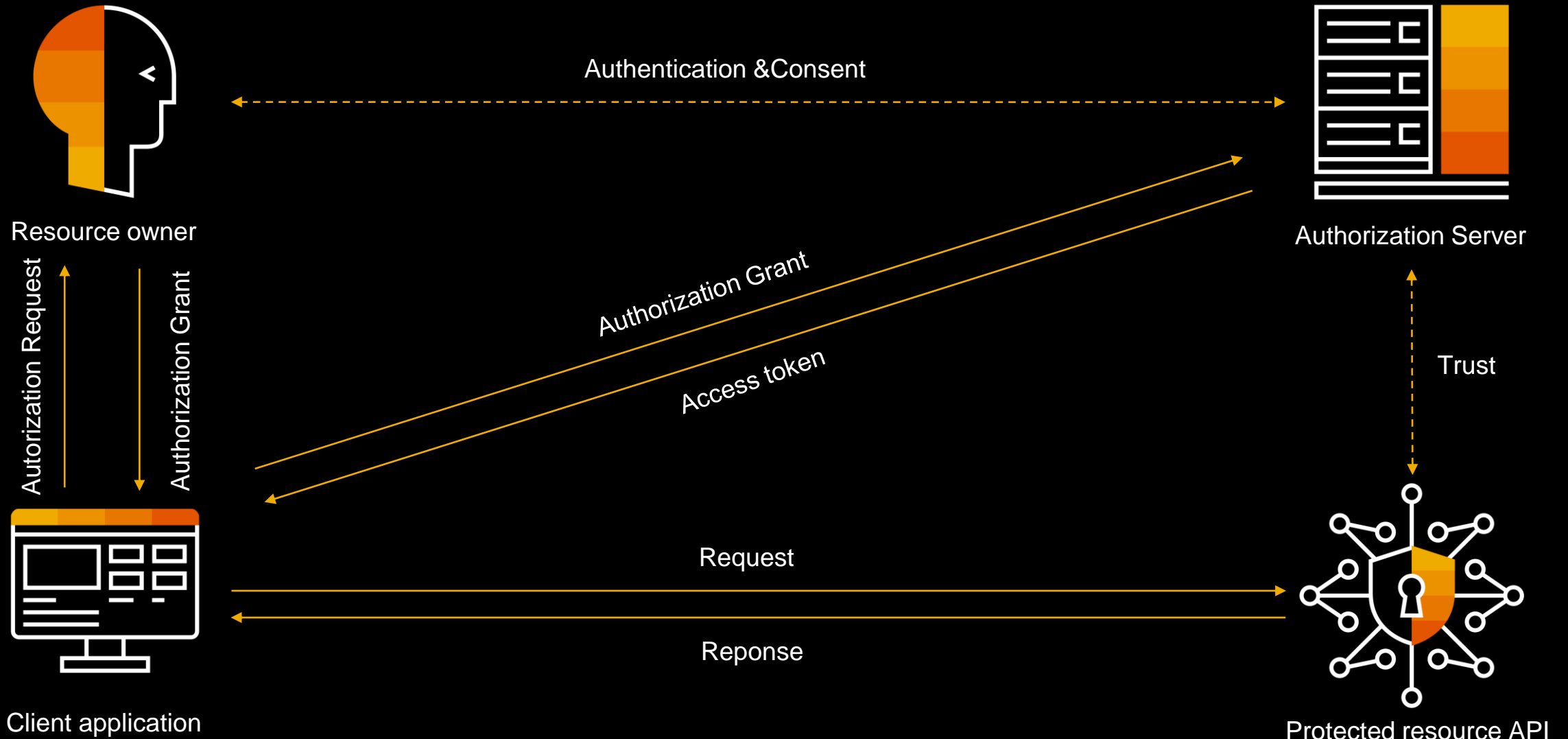


Server autentykacji
Authorization server



Strzeżony zasób (API)
Protected resource

OAUTH FLOW 2.0





DEMO

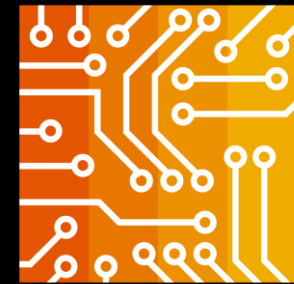
OAUTH 2.0 nieporozumienia



Access Token nie reprezentuje
użytkownika użytkownika

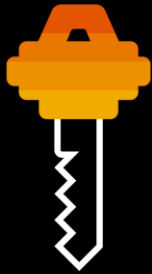


Access Token jest przeznaczony
dla strzeżonego zasobu (API)



Aplikacja kliencka nie może
Polegać na zawartości Access Tokenu

OAUTH 2.0



Format Access Tokenu



Walidacja Access Tokenu



Autentykacja użytkownika

Zalety OAUTH 2.0



Delegowanie dostępu



Stworzony z myślą o
zabezpieczeniu API



Zgoda użytkownika

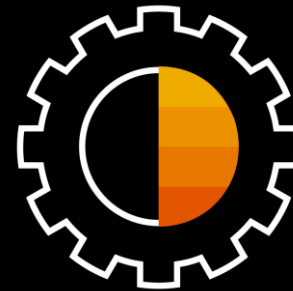


Autoryzacja użytkownika i klienta
jest rozdzielona

OAuth2 Endpoints



Authorization



Token

Oauth 2 Scopes

HYC_API

HYC_API.read

HYC_API.lectures

Authorization code flow

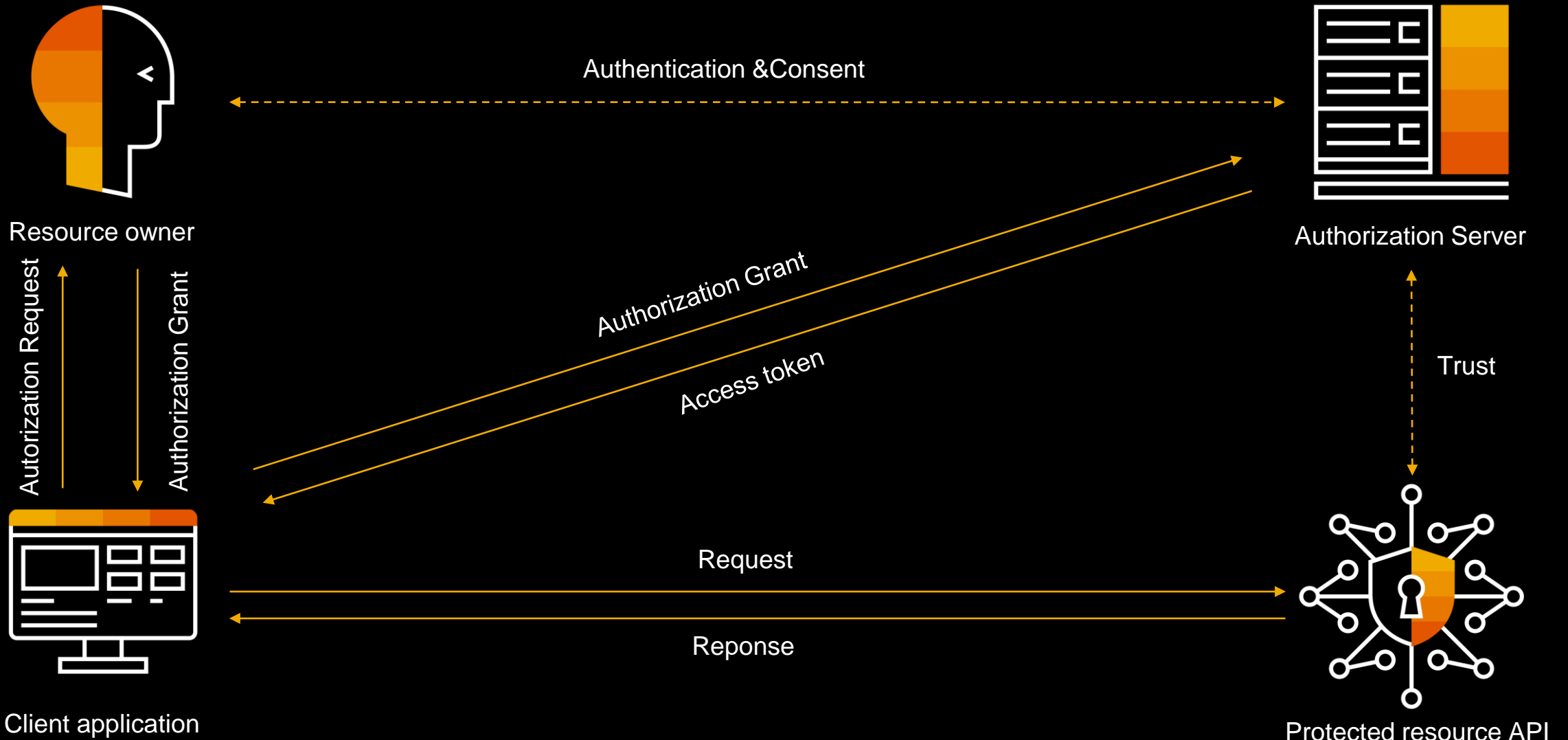


Zaprojektowany dla "poufnych klientów"



Najlepszy dla aplikacji, które posiadają backend

Authorization code flow



Authorization request

https://auth.server.com/authorize

?response_type=code

&client_id=Lkc11D1kzz

&redirect_uri=https://client.app.com/callback

&scope=hyc_api.feed.read hyc_api.lectures.read

&state=dNjDVX9qfs

Authorization response

https://client.app.com/callback

?code=SpIxIOBeZQQYbYS6WxSbIA

&state=dNjDVX9qfs

Token request

POST /token HTTP/1.1

Host: auth.server.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code

&code=SpIxlOBeZQQYbYS6WxSbIA

&redirect_uri=https://client.app.com/callback

Basic authentication

RFC 7617

Bas64(client_id + ":" + client_secret)

Token response

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

```
{  
  "access_token" : "2YotnFZFEjr1zCsicMWpAA",  
  "token_type" : "example",  
  "expires_in" : 3600  
}
```



DEMO

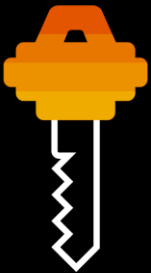
Implicit flow



Zaprojektowany dla "publicznych klientów"

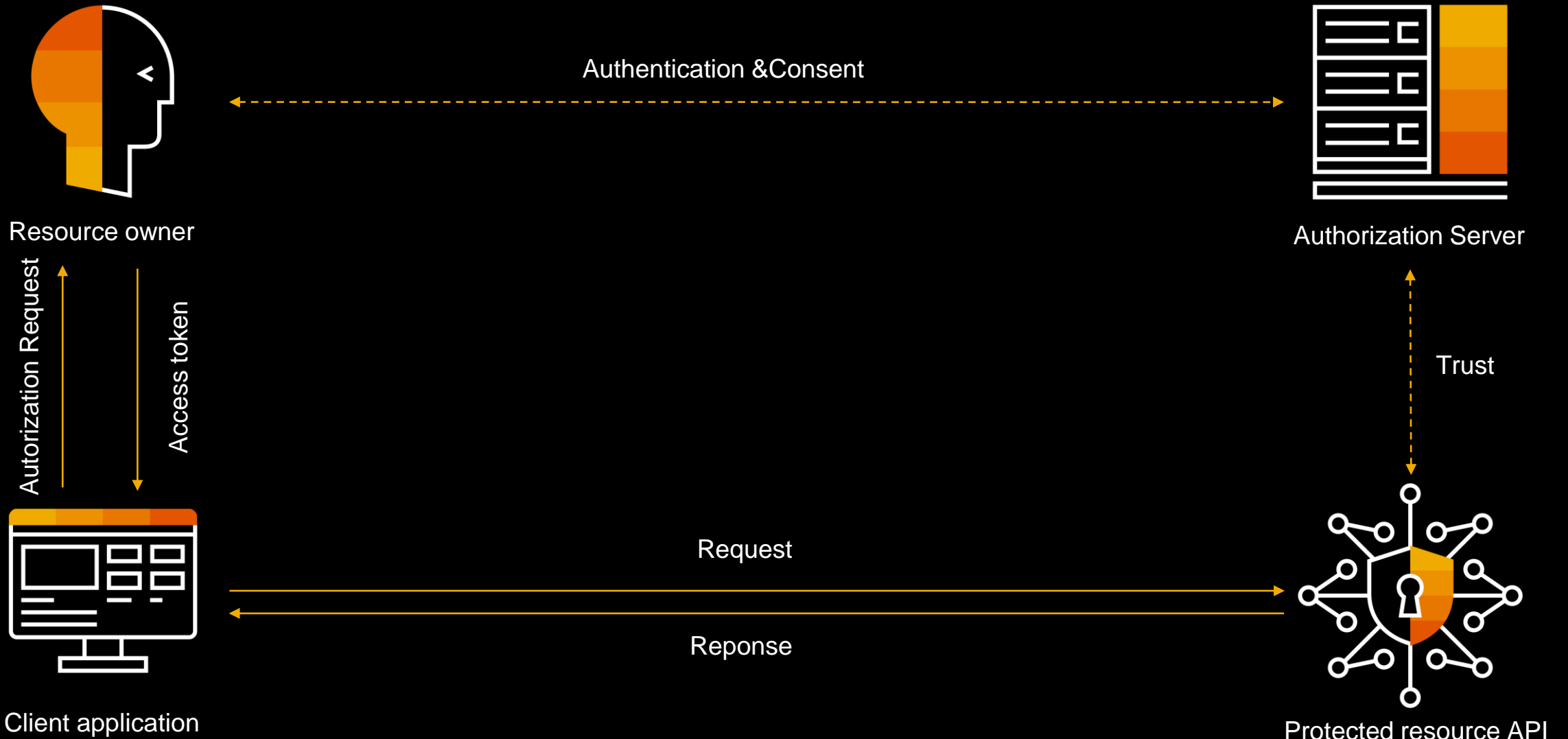


Najlepszy dla aplikacji, które działają w przeglądarce



Brak autentykacji klienta

Implicit flow



Authorization request

https://auth.server.com/authorize

?response_type=token

&client_id=Lkc11D1kzz

&redirect_uri=https://client.app.com/callback

&scope=hyc_api.feed.read hyc_api.lectures.read

&state=dNjDVX9qfs

Authorization response

https://client.app.com/callback

#access_token=SpIxlOBeZQQYbYS6WxSbIA

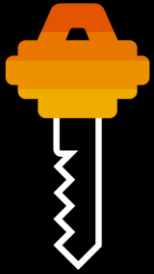
&token_type=example

&expires_in=3600

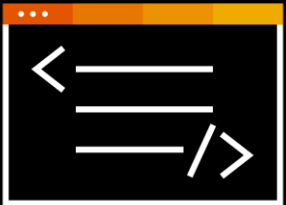
&scope=hyc_api.feed.read hyc_api.lectures.read

&state=dNjDVX9qfs

Obawy dotyczące bezpieczeństwa



Access Token dostępne są dla „Resource ownera”



Access Token dostępne jest dla 3rd part JS lib



Brak walidacji odnośnie dla kogo przeznaczony jest Token



DEMO

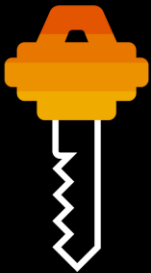
Client credentials flow



Zaprojektowany dla aplikacji, które są "Resource ownerem"

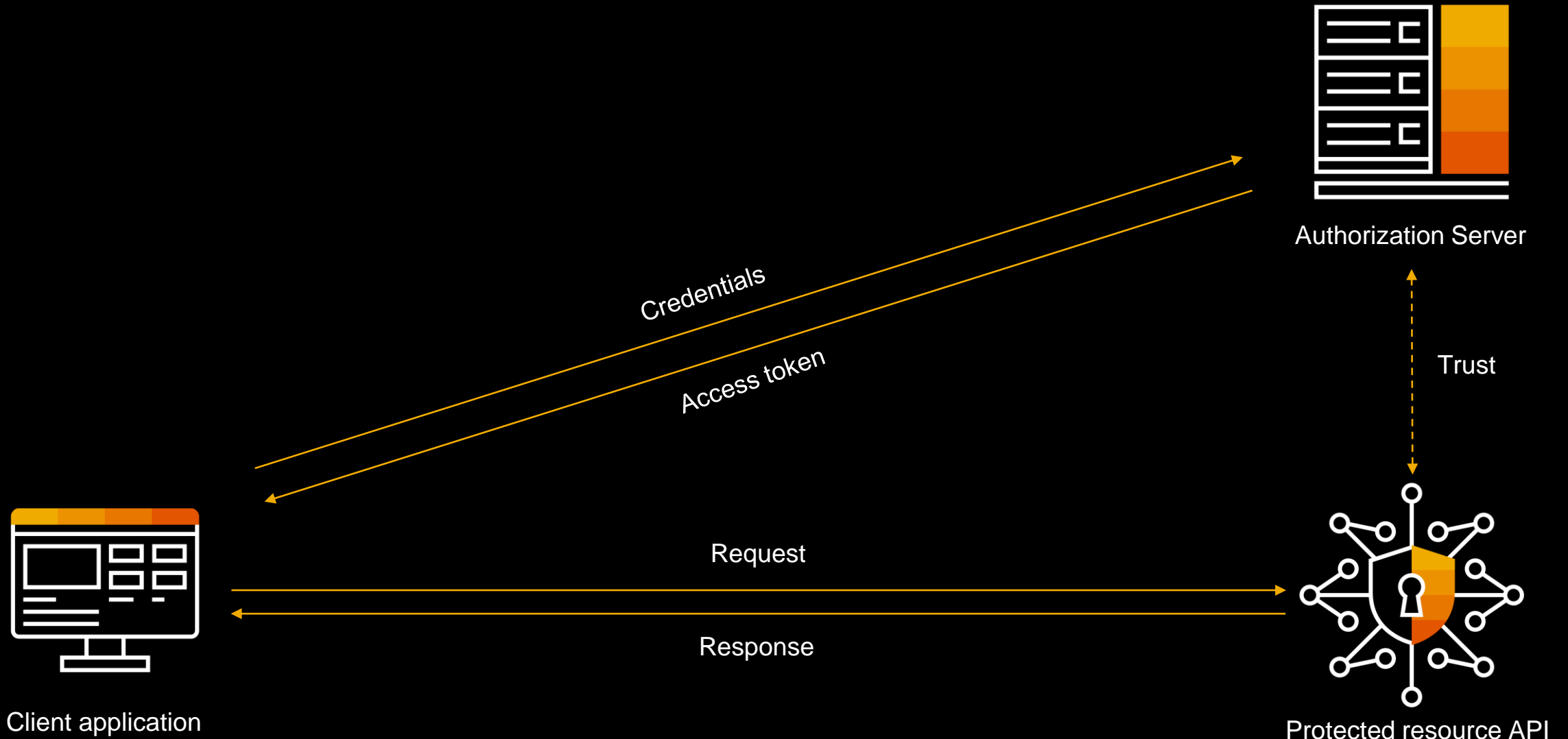


Najlepszy dla komunikacji backend-to-backend



Wymaga autentykacji klienta

Client credentials flow



Token request

POST /token HTTP/1.1

Host: auth.server.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials

&scope=hyc_api.feed.read hyc_api.lectures.read

Token response

HTTP/1.1 200 OK

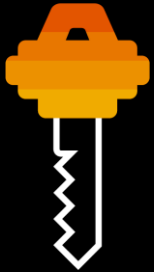
Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

```
{  
  "access_token" : "2YotnFZFEjr1zCsicMWpAA",  
  "token_type" : "example",  
  "expires_in" : 3600  
}
```

Różnica pomiędzy Access Tokenem a HTTP basic authentication



Nie wysyłamy danych do logowania przy każdym zapytaniu



Access Token ma swoją datę ważności



DEMO

ROPC flow



Zaprojektowany do łatwiejszej migracji przestarzałych aplikacji



Neguje większość założeń OAuth



Nie powinien być używany

Token request

POST /token HTTP/1.1

Host: auth.server.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant_type=password

&username=usename&password=pa\$\$word

&scope=hyc_api.feed.read hyc_api.lectures.read

Token response

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

```
{  
    "access_token" : "2YotnFZFEjr1zCsicMWpAA",  
    "token_type" : "example",  
    "expires_in" : 3600  
}
```



DEMO

Refresh token



Wymiana na
nowy Token



Pozwala na
długotrwałe działanie aplikacji



Jest niezwykle
poufny



Użytkownik zawsze
powinien zostać
poinformowany
o jego dostępności

Authorization request

https://auth.server.com/authorize

?response_type=code

&client_id=Lkc11D1kzz

&redirect_uri=https://client.app.com/callback

&scope=hyc_api.feed.read **offline_access**

&state=dNjDVX9qfs

Token response

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

```
{  
  "access_token" : "2YotnFZFEjr1zCsicMWpAA",  
  "token_type" : "example",  
  "expires_in" : 3600,  
  "refresh_token" : ">2L/%X28N]BZ%Ps]"
```

Token request

POST /token HTTP/1.1

Host: auth.server.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

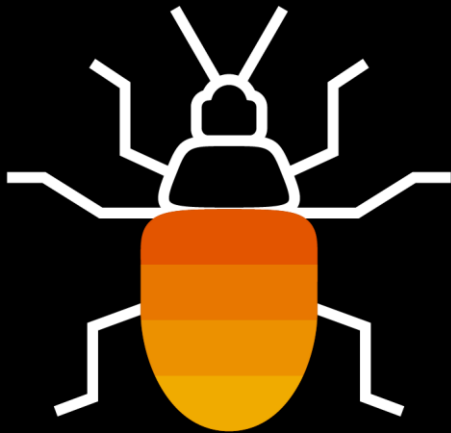
grant_type=refresh_token

&scope=hyc_api.feed.read hyc_api.lectures.read

Refresh token

Authorization flow	Może dostać refren token
Authorization code	Tak
Implicit	Nie
Client credentials	Tak
ROPC	Nie

Obsługa błędów



- **error**
- **error_descirption**
- **error_uri**
- **state**

Obsługa błędów

`https://auth.server.com/authorize`
`?error=invalid_request`
`&state=dNjDVX9qfs`

Obsługa błędów

HTTP/1.1 400 Bad Request

Content-Type: application/json;charset=UTF-8

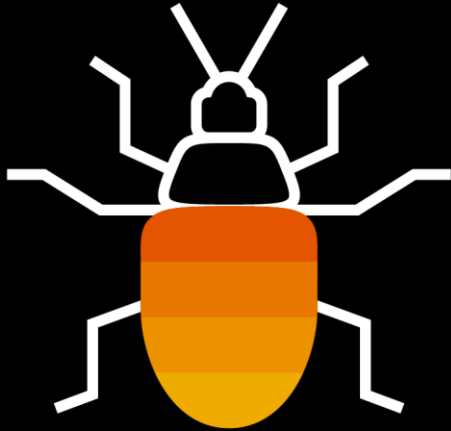
Cache-Control: no-store Pragma: no-cache

{

"error": "invalid_request"


}

Typy błędów



- **invalid_request**
- **invalid_client (401)**
- **invalid_grant**
- **unauthorized_client**
- **unsupported_grant_type**
- **invalid_scope**

Co dalej?

- 
- **Open ID Connect**
 - **RFC 7636: Proof Key for Code Exchange**
 - **RFC 8252: OAuth 2.0 for Mobile and Native Apps**
 - **RFC 8626: OAuth 2.0 Device Authorization Grant**
 - **RFC 8414: Authorization Server Metadata**
 - **RFC 6819: OAuth 2.0 Threat Model and Security Considerations**
 - **RFC 7522: Security Assertion Markup Language (SAML) 2.0**

**Profile for OAuth 2.0 Client Authentication
and Authorization Grants**

Pytania?

Thank you.

Diploma Student / Intern Engineer Job
Apply now:

