

# **Security Advanced– PE Opdracht**

**Opdracht Security Advanced – Yellow Team aspect:**

**SSO met OAuth2 en OpenID Connect**

**Github repository:**

[https://github.com/janvinkenroyePXL/secadv\\_pe\\_yellowteam](https://github.com/janvinkenroyePXL/secadv_pe_yellowteam)

## Inhoud

Gegevens van API ophalen via Postman (Level C).....	4
Stap 1: Configuratie van de STS-server opvragen .....	4
Stap 2: Access token opvragen.....	5
Stap 3: API aanspreken met access token.....	5
Gegevens van API ophalen via client (Level C) .....	6
Inspectie access token via jwt.io (Level C) .....	9
Opvragen JWKS in Postman .....	9
Decoderen JWT op jwt.io .....	10
Verifiëren signature.....	10
Access token aanpassen (Level C) .....	13
Veranderen encryptie.....	13
Access token met aangepaste issuer.....	13
Access token met aangepaste client id .....	13
Access token met aangepaste scope.....	13
Evaluatie van de implementatie van de authorization code flow door de API (Level C) .....	14
Web API die gebruik maakt van Auth0 (Level B) .....	15
Controllers .....	15
appsettings.json .....	17
ConfigureServices .....	17
HasScopeRequirement en HasScopeHandler.....	18
Configure .....	20
Web App die de Web API aanspreekt (Level B) .....	21
Configure .....	21
appsettings.json .....	21
ApiAccessClient .....	21
ConfigureServices .....	23
HomeController.....	23
Web App die Auth0 en OpenID Connect gebruikt en de Web API aanspreekt (Level A) .....	25
Configure .....	25
appsettings.json .....	25
ConfigureServices .....	25
AccountController .....	29

StudentsController en PoemController .....	29
Two-Factor Authentication (Level A+).....	32
Social connections (Level A+) .....	37

# Gegevens van API ophalen via Postman (Level C)

## Stap 1: Configuratie van de STS-server opvragen

security-advanced-c / sts-server-discovery

GET https://ventielshop.dubbabub.be:8081/.well-known/openid-configuration ... Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
-----	-------	-------------

Body Cookies Headers (4) Test Results Status: 200 OK Time: 249 ms Size: 1.71 KB Save Response

Pretty Raw Preview Visualize JSON Copy

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
{
  "issuer": "https://ventielshop.dubbabub.be:8081",
  "jwks_uri": "https://ventielshop.dubbabub.be:8081/.well-known/openid-configuration/jwks",
  "authorization_endpoint": "https://ventielshop.dubbabub.be:8081/connect/authorize",
  "token_endpoint": "https://ventielshop.dubbabub.be:8081/connect/token",
  "userinfo_endpoint": "https://ventielshop.dubbabub.be:8081/connect/userinfo",
  "end_session_endpoint": "https://ventielshop.dubbabub.be:8081/connect/endsession",
  "check_session_iframe": "https://ventielshop.dubbabub.be:8081/connect/checksession",
  "revocation_endpoint": "https://ventielshop.dubbabub.be:8081/connect/revocation",
  "introspection_endpoint": "https://ventielshop.dubbabub.be:8081/connect/introspect",
  "device_authorization_endpoint": "https://ventielshop.dubbabub.be:8081/connect/deviceauthorization",
  "frontchannel_logout_supported": true,
  "frontchannel_logout_session_supported": true,
  "backchannel_logout_supported": true,
  "backchannel_logout_session_supported": true,
  "scopes_supported": [
    "api",
    "offline_access"
  ],
  "claims_supported": [],
  "grant_types_supported": [
    "authorization_code",
    "client_credentials",
    "refresh_token",
    "implicit",
    "urn:ietf:params:oauth:grant-type:device_code"
  ],
  "response_types_supported": [
    "code",
    "token",
    "id_token",
    "id_token token",
    "code id_token",
```

## Stap 2: Access token opvragen

security-advanced-c / sts-server-token

POST https://ventielshop.dubbudub.be:8081/connect/token

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
grant_type	client_credentials	
client_id	pxl-secadv	
client_secret	maarten_just_geen_spruitjes	
scope	api1	
Key	Value	Description

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 201 ms Size: 965 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZWQzIiwiaWF0Ij0iMTY2Zm50X2liIjoicHhsLXNlY2FkdIsImp0aSI6Im5RUYxRTlFMzBFNDk3MTc2NjMzNTIxQjhmMjJDNzZBIiwiaWF0Ij0iMTY2Zm50X2liIj0i",
3   "expires_in": 3600,
4   "token_type": "Bearer",
5   "scope": "api1"
6 }
```

## Stap 3: API aanspreken met access token

security-advanced-c / web-api

GET https://ventielshop.dubbudub.be/fiets

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to Request Headers

Current Token

This access token is only available to you. Sync the token to let collaborators on this request use it.

Access Token

Available Tokens

eyJhbGciOiJIUzI1NiIsImtpZCI6IkpXZWQzIiwiaWF0Ij0iMTY2Zm50X2liIjoicHhsLXNlY2FkdIsImp0aSI6Im5RUYxRTlFMzBFNDk3MTc2NjMzNTIxQjhmMjJDNzZBIiwiaWF0Ij0iMTY2Zm50X2liIj0i

Header Prefix

Bearer

Status: 200 OK Time: 168 ms Size: 593 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "Het Hollands ventieldopje (Dunlop) is een veel voorkomend ventiel. Vaak zie je deze terug op een stadsfiets. Ze zijn eenvoudig op te pompen.",
3   "Het Frans ventieldopje komt vaak voor bij sportieve fietsen zoals racefietsen. Dit ventieldopje is al iets moeilijker om op te pompen.",
4   "Het Auto ventieldopje is een populair ventieldopje, voornamelijk in de auto branche. Het lijkt er op dat dit type ventieldopje steeds meer verdwijnt bij fietsen."
5 }
```

## Gegevens van API ophalen via client (Level C)

Ik heb een .Net Core consoleprogramma geschreven (te vinden in de Githubrepo onder de naam consoleapp). Hieronder volgt de code van Program.cs.

```
using Newtonsoft.Json;
using RestSharp;
using System;

namespace consoleapp
{
    class Program
    {
        static void Main(string[] args)
        {
            string accessToken = GetToken();
            string[] response = CallApi(accessToken);
            foreach (string line in response)
            {
                Console.WriteLine(line);
            }
        }

        private static string GetToken()
        {
            var client = new
RestClient("https://ventielshop.dubbadub.be:8081/connect/token");
            // Bypass SSL validation
            client.RemoteCertificateValidationCallback = (sender, certificate, chain,
sslPolicyErrors) => true;
            client.Timeout = -1;
            var request = new RestRequest(Method.POST);
            request.AddHeader("content-type", "application/x-www-form-urlencoded");
            request.AddHeader("Authorization", "Basic Og==");
```

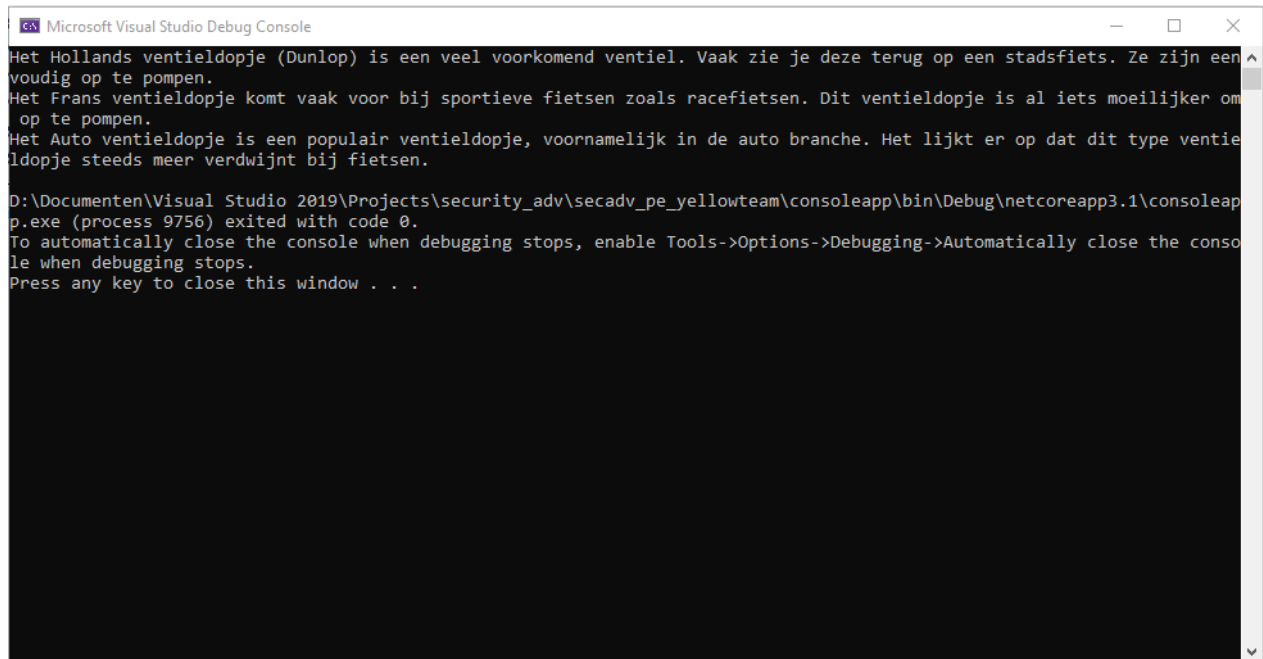
```

        request.AddParameter("grant_type", "client_credentials");
        request.AddParameter("client_id", "pxl-secadv");
        request.AddParameter("client_secret", "maarten_lust_geen_spruitjes");
        request.AddParameter("audience", "");
        IRestResponse response = client.Execute(request);
        dynamic jsonResponse = JsonConvert.DeserializeObject(response.Content);
        return jsonResponse.access_token;
    }

    private static string[] CallApi(string accessToken)
    {
        var client = new RestClient("https://ventielshop.dubbadub.be/fiets");
        // Bypass SSL validation
        client.RemoteCertificateValidationCallback = (sender, certificate, chain,
sslPolicyErrors) => true;
        client.Timeout = -1;
        var request = new RestRequest(Method.GET);
        request.AddHeader("Authorization", "Bearer " + accessToken);
        IRestResponse response = client.Execute(request);
        dynamic jsonResponse = JsonConvert.DeserializeObject(response.Content);
        return jsonResponse.ToObject<string[]>();
    }
}
}

```

Als we het programma runnen krijgen we de volgende output in de console:



```
Microsoft Visual Studio Debug Console
Het Hollands ventieldopje (Dunlop) is een veel voorkomend ventiel. Vaak zie je deze terug op een stadsfiets. Ze zijn een
voudig op te pompen.
Het Frans ventieldopje komt vaak voor bij sportieve fietsen zoals racefietsen. Dit ventieldopje is al iets moeilijker om
op te pompen.
Het Auto ventieldopje is een populair ventieldopje, voornamelijk in de auto branche. Het lijkt er op dat dit type ventie
ldopje steeds meer verdwijnt bij fietsen.

D:\Documenten\Visual Studio 2019\Projects\security_adv\secadv_pe_yellowteam\consoleapp\bin\Debug\netcoreapp3.1\consoleapp
p.exe (process 9756) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```



# Inspectie access token via jwt.io (Level C)

## Opvragen JWKS in Postman

security-advanced-c / sts-server-jwks-endpoint

GET https://ventielshop.dubbadub.be:8081/well-known/openid-configuration/jwks

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (4) Test Results

Status: 200 OK Time: 200 ms Size: 599 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "keys": [
3     {
4       "kty": "RSA",
5       "use": "sig",
6       "kid": "DC8BD67B968D0C9C999668EF7482D081D3",
7       "e": "AQAB",
8       "n": "wVbhTsNBETlW3M62AdWbmzNnZYpPnyAtjE9SEoxTmUkSHOIXu1Qczo9D9_5hfYEZwN0sYJZI4gqVMb6fSp_FRQ68qeNksx_111kC0c5x81SjFjuYnE108a00j-T9wTwDcd3ZlQej1v6Zz-M6u1AuID6t72NC19S8QA3vgMkHE72nN-aLnXFDu3p3pt-7NaIoMQ06ci_8KI0D0WjWzLcA5A6PF1UL-JKpyUj5-aQWm9SsTn0xVs380Qht8_cxvk56e5b4X5kwC7pdP0vYnyquJyzQsejQ6UvynnggZNg1Hca08_P6nEABF63yCndohwJE_xaXs6Vxj_dikeBDGn3gsVg4w",
9       "alg": "RS256"
10     }
11   ]
12 }
```

## Decoderen JWT op jwt.io

Algorithm RS256

Encoded PASTE A TOKEN HERE

Decoded EDIT THE PAYLOAD AND SECRET

```
eyJhbGciOiJSUzI1NiIsImtpZCI6Ikd0EJENjdCOTY4RD
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "kid": "DC8BD67B968D0C9C99B68EF74B2D01D3",
  "typ": "at+jwt"
}
```

PAYLOAD: DATA

```
{
  "nbf": 1622025105,
  "exp": 1622028705,
  "iss": "https://ventielshop.dubbadub.be:8081",
  "client_id": "pxl-secadv",
  "jti": "C9EF1E9E30E497176633521B8A22C76A",
  "iat": 1622025105,
  "scope": [
    "api1"
  ]
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key or Certificate. Enter it in plain text only if you want to verify a token
```

## Verifiëren signature

We stellen allereerst vast dat de property 'alg' in de header van de gedecodeerde jwt dezelfde is dan de property 'id\_token\_signing\_alg\_values\_supported' in de response van het well-known-endpoint van de STS-server.

We kunnen ook vaststellen dat de property 'kid' in de header van de gedecodeerde jwt gelijk is aan de 'kid' property van de enige key in de jwks. Dit betekent dat we deze jwk kunnen gebruiken om de signature te verifiëren.

Aangezien de JWK geen 'x5c' property bevat, moeten we de signature verifiëren met behulp van de waarden van de properties 'e' (exponent) en 'n' (modulus).

Om de public key te bekomen voeren we het volgende uit.

```
<?php
include('Crypt/RSA.php');
include('Math/BigInteger.php');
```

```

$exponent = 'AQAB';

$modulus =
'wVbhTsNBETlW3M62AdWbmzNnZYpPnyAtjE0SEorTmUkSH0IxuiQCro9D9_5hfYEZwN0sYJZI4gqVMB6fSp_f
RQG5qeNksx_I11kCDc5rB1SjFjuYnE10Ba00j-T9wTWDcd3ZlQejlv5Zz-
M6uiAuID5t72NCi9S8QA3vgMKhE72nN-aLnXfDU3p3pt-7NaIoMQ06ci_8KIDDWjWzLcA5A6PF1UL-
JKpyUj5-
aQWM9SsTn0xVs380QHt8_cxvk56e5b4X5kwC7pdP0vYnyquJYzQsjQ5UvynggZNg1Hca08_P6nEABF53yCnDd
hWjE_xaXs6Vxj_diKeBDGn3gsVgAw';

$rsa = new Crypt_RSA();
$rsa->loadKey([
    'e' => new Math_BigInteger(base64_decode($exponent), 256),
    'n' => new Math_BigInteger(base64_decode($modulus), 256)
]);

echo $rsa;

```

We kunnen dan de signature verifiëren op [jwt.io](https://jwt.io) met behulp van deze public key.

```

Header:
{
  "alg": "RS256",
  "kid": "DC8BD67B968D0C9C99B68EF74B2D01D3",
  "typ": "at+jwt"
}
Payload:
{
  "nbf": 1621598849,
  "exp": 1621602449,
  "iss": "https://ventielshop.dubbadub.be:8081",
  "client_id": "pxl-secadv",
  "jti": "CFC5C7B32BF0474C64FA4D5F3A538103",
  "iat": 1621598849,
  "scope": [
    "api1"
  ]
}

```

```
]
}
Signature:
RSASHA256(
    base64UrlEncode(header) + "." +
    base64UrlEncode(payload),

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0NFVHPrjikIbAK7MaVJH
nn8Hl7Wz6qQEa/bsrFGwPH90kQrG34HY9IL46iCvIeQnTBOT2W22c1lPPuvB/Sek
m2TtTs2W6Ial09n1ZwZiBIQGtKFCKZ03413c9hAFXtxEurFqHfhmqMkjbS3t/fGF
Q2i6SE3eilc9e5vTFUGY/5MzBuXGMscpJL/UeThS3JPmrMbWPRpmczivN5gi9is0
qrZ8dYxRGgaqfM91V+NpGLigSIMuYiPd5CvCpMOJQBRbnw6cM6EJdzhGc6C6ef0z
/Ec0EWPa+dAOvE3tchFJNBpHqNdQD6hjeK3PT0zQsxqW85IErCJqBb9ubJcSQCVn
mQIDAQAB
-----END PUBLIC KEY-----
```



## Evaluatie van de implementatie van de authorization code flow door de API (Level C)

- Geen geldig SSL-certificaat
- De audience wordt niet gevalideerd. Daardoor ontstaat er een vulnerability voor forwarding attacks.
- De environment is 'Development' en er wordt bijgevolg een DeveloperExceptionPage gebruikt. Dit is een risico aangezien gedetailleerde informatie over excepties bevat die gebruikt kan worden door een aanvaller
- Er wordt geen gebruik gemaakt van scopes of policies bij de authorisatie
- Er wordt geen Authority parameter ingesteld bij het configureren van de JwtBearer

## Web API die gebruik maakt van Auth0 (Level B)

Zie de webapi op Github.

### Controllers

De ***StudentsController*** bevat een aantal endpoints, waaronder het endpoint */api/students* dat de leden van ons team teruggeeft na een succesvolle GET request. Dit endpoint gebruikt de autorisatiepolicy *team*.

```
public class StudentsController : ControllerBase
{
    private IStudentData _studentData;
    public StudentsController(IStudentData studentData)
    {
        _studentData = studentData;
    }

    [HttpGet]
    [Route("")]
    [Route("All")]
    [Authorize("team")]
    public IActionResult Get()
    {
        var model = _studentData.GetAll();
        return new ObjectResult(model);
    }
}
```

*Geïmplementeerde endpoints:*

<https://localhost:44339/api/Students>

<https://localhost:44339/api/Students/1>

<https://localhost:44339/api/Students/1/update>

<https://localhost:44339/api/Students/1/delete>

<https://localhost:44339/api/Students/new>

De **PoemsController** bevat twee endpoints:

- het endpoint `/api/Poems` dat alle gedichtjes teruggeeft.
- het endpoint `/api/Poems/{id?}` dat een gedichtje met een bepaalde id teruggeeft.

Beide endpoints gebruiken de autorisatiepolicy *admin*

```
[Route("api/[controller]")]
public class PoemsController : ControllerBase
{
    private IPoemData _poemData;

    public PoemsController(IPoemData poemData)
    {
        _poemData = poemData;
    }

    [HttpGet]
    [Authorize("admin")]
    public IActionResult Get()
    {
        var model = _poemData.GetAll();
        return new ObjectResult(model);
    }

    [HttpGet]
    [Route("{id?}")]
    [Authorize("admin")]
    public IActionResult GetById(int id)
    {
        var model = _poemData.Get(id);
        if(model == null)
        {
            return NotFound();
        }
    }
}
```



```

        }
        else
        {
            return new ObjectResult(model);
        }
    }
}
}

```

*Geïmplementeerde endpoints:*

<https://localhost:44339/api/Poems>

<https://localhost:44339/api/Poems/1>

## appsettings.json

De appsettings.json bevat volgende object-initialiser met de gegevens van ons Auth0 Domain en onze Auth0 API Identifier.

```

"Auth0": {
    "Domain": "secadvpe-dev.eu.auth0.com",
    "Audience": "https://localhost:44339"
}

```

## ConfigureServices

In de *ConfigureServices* methode van de Startup klasse worden de services *AddAuthentication* en *AddJwtBearer* geconfigureerd. In *AddAuthentication* worden de authenticatieschema's ingesteld. Wij gebruiken het *JwtBearerDefaults.AuthenticationScheme* van de Nuget package *Microsoft.AspNetCore.Authentication.JwtBearer*. In de *AddJwtBearer* voeren we de gegevens van onze Authority en onze Audience in.

```

var domain = $"https://{Configuration["Auth0:Domain"]}";
services.AddAuthentication(options =>
{
    options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
}
)

```

```

options.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme;
options.DefaultSignOutScheme = JwtBearerDefaults.AuthenticationScheme;
options.DefaultForbidScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(options =>
{
options.Authority = "https://secadvpe-dev.eu.auth0.com/";
options.Audience = "https://localhost:44339";
});

```

Verder configureren we de service *AddAuthorization*. We maken gebruik van policy based authentication. Voor elke scope die we hebben aangemaakt in Auth0 voegen we een policy toe. De policy bestaat er telkens eenvoudig in dat de betreffende scope moet zijn toegekend.

```

services.AddAuthorization(options =>
{
options.AddPolicy("admin", policy => policy.Requirements.Add(new
HasScopeRequirement("admin", domain)));

options.AddPolicy("team", policy => policy.Requirements.Add(new
HasScopeRequirement("team", domain)));

options.AddPolicy("team-crud", policy => policy.Requirements.Add(new
HasScopeRequirement("team-crud", domain)));
});

services.AddSingleton<IAuthorizationHandler, HasScopeHandler>();

```

## HasScopeRequirement en HasScopeHandler

We maken een klasse *HasScopeRequirement*, die de gegevens over een scope bevat. Deze klasse implementeert de interface *IAuthorizationRequirement*.

```

public class HasScopeRequirement : IAuthorizationRequirement
{
public string Issuer { get; }
public string Scope { get; }

public HasScopeRequirement(string scope, string issuer)
{

```

```

        Scope = scope ?? throw new ArgumentNullException(nameof(scope));
        Issuer = issuer ?? throw new ArgumentNullException(nameof(issuer));
    }
}

```

We maken ook een klasse *HasScopeHandler*.

```

public class HasScopeHandler : AuthorizationHandler<HasScopeRequirement>
{
    protected override Task HandleRequirementAsync(AuthorizationHandlerContext
context, HasScopeRequirement requirement)
    {
        // If user does not have the scope claim, get out of here
        if (!context.User.HasClaim(c => c.Type == "scope" && c.Issuer ==
requirement.Issuer))
            return Task.CompletedTask;

        // Split the scopes string into an array
        var scopes = context.User.FindFirst(c => c.Type == "scope" && c.Issuer ==
requirement.Issuer).Value.Split(' ');

        // Succeed if the scope array contains the required scope
        if (scopes.Any(s => s == requirement.Scope))
            context.Succeed(requirement);

        return Task.CompletedTask;
    }
}

```

Tenslotte registreren we deze service als een singleton in *ConfigureServices*.

```

services.AddSingleton<IAuthorizationHandler, HasScopeHandler>();

```

## Configure

In de *Configure* methode van de Startup klasse roepen we de *UseAuthentication* en *UseAuthorization* methoden aan. We roepen ook *UseHttpsRedirection* aan om http requests om te leiden naar https.

```
app.UseHttpsRedirection();  
app.UseAuthentication();  
app.UseAuthorization();
```

## Web App die de Web API aanspreekt (Level B)

Zie de webapp1 op Github. Deze web app gebruikt de client credentials flow om gegevens van de Web API op te halen.

### Configure

In de *Configure* methode van de Startup klasse roepen we *UseHttpsRedirection* aan om http requests om te leiden naar https.

```
app.UseHttpsRedirection();
```

### appsettings.json

De appsettings.json bevat volgende object-initialiser met de gegevens van ons Auth0 Domain, onze Auth0 API Identifier, de ClientId en de ClientSecret van onze webapp1.

```
"Auth0": {  
  "Domain": "secadvpe-dev.eu.auth0.com",  
  "ClientId": "atPLIdzEnjhuMtizu6x64EEE5FJvXTJg",  
  "ClientSecret":  
    "WoeNSEKNpql5ChvVyiTEMoGQy5lXCLJUzVRxuI09zzSvLFzbswYZcB70bNdXyGgm",  
  "ApiIdentifier": "https://localhost:44339"  
}
```

### ApiAccessClient

We maken een interface IApiAccessClient en een klasse ApiAccessClient die deze interface implementeert.

We injecteren de configuratie in de constructor zodat we de gegevens uit onze appsettings.json kunnen opslaan in velden.

```
private readonly string ClientSecret;  
private readonly string ClientId;  
private readonly string Domain;  
private readonly string ApiIdentifier;  
public string AccessToken { get; private set; }  
private static RestClient Client;
```

```

public ApiAccessClient(IConfiguration configuration)
{
    ClientSecret = configuration["Auth0:ClientSecret"];
    ClientId = configuration["Auth0:ClientId"];
    Domain = configuration["Auth0:Domain"];
    ApiIdentifier = configuration["Auth0:ApiIdentifier"];
    Client = new RestClient("https://" + Domain + "/oauth/token");
    Authenticate();
}

```

We maken in deze interface/klasse een methode Authenticate om een access token op te vragen aan onze STS.

```

public void Authenticate()
{
    var request = new RestRequest(Method.POST);
    request.AddHeader("content-type", "application/x-www-form-urlencoded");
    request.AddParameter("grant_type", "client_credentials");
    request.AddParameter("client_id", ClientId);
    request.AddParameter("client_secret", ClientSecret);
    request.AddParameter("audience", ApiIdentifier);
    IRestResponse response = Client.Execute(request);
    dynamic jsonResponse = JsonConvert.DeserializeObject(response.Content);
    AccessToken = jsonResponse.access_token;
}

```

We maken ook een methode CallApi aan om de eigenlijke gegevens op te vragen van onze Web API.

```

public IRestResponse CallApi(string endpoint)
{
    if(AccessToken == null || AccessToken == "Not found.")
    {
        Authenticate();
    }
}

```

```

    var client = new RestClient(ApiIdentifier + "/api/" + endpoint);
    var request = new RestRequest(Method.GET);
    request.AddHeader("content-type", "application/json");
    request.AddHeader("authorization", "Bearer " + AccessToken);
    IRestResponse response = client.Execute(request);
    return response;
}

```

## ConfigureServices

We registreren de service *ApiAccessClient* als *Scoped* in de *ConfigureServices* methode van de *Startup* klasse.

```

public void ConfigureServices(IServiceCollection services)
{
    services.AddScoped<IApiAccessClient, ApiAccessClient>();
    services.AddControllersWithViews();
}

```

## HomeController

De **HomeController** bevat een aantal routes met daaraan gekoppelde acties.

De route `/students` maakt gebruik van de `ApiAccessClient.CallApi` methode om request te doen naar het `/api/students` endpoint van onze Web API. De JSON-data uit de response worden via een model omgezet in een view.

```

[Route("Students")]
public IActionResult Students()
{
    IRestResponse response = _accessClient.CallApi("students");
    if (response.IsSuccessful)
    {
        var model = response.Content;
        List<StudentViewModel> data =
        JsonConvert.DeserializeObject<List<StudentViewModel>>(model);
        return View(data);
    }
}

```

```

        else
        {
            throw new Exception("Unable to get content");
        }
    }
}

```

De route /poem doet een request naar het /api/poems endpoint van onze Web API. Er wordt ook weer gebruik gemaakt van de ApiAccessClient service. Eén willekeurig gedicht uit de response wordt via een model omgezet in een View.

```

[Route("[action]")]
public IActionResult Poem()
{
    IRestResponse response = _accessClient.CallApi("poems");
    if (response.IsSuccessful)
    {
        var model = response.Content;
        List<PoemViewModel> poemsList =
        JsonConvert.DeserializeObject<List<PoemViewModel>>(model);
        Random random = new Random();
        PoemViewModel data = poemsList[random.Next(poemsList.Count)];
        return View(data);
    }
    else
    {
        throw new Exception("Unable to get content");
    }
}

```

*Geïmplementeerde pagina's:*

<https://localhost:44357>

<https://localhost:44357/Students>

<https://localhost:44357/Poem>



## Web App die Auth0 en OpenID Connect gebruikt en de Web API aanspreekt (Level A)

Zie de webapp2 op Github. Deze web app gebruikt de authorization code flow om gegevens van de Web API op te halen.

### Configure

In de *Configure* methode van de Startup klasse roepen we opnieuw de *UseAuthentication* en *UseAuthorization* methoden aan. We roepen ook *UseHttpsRedirection* aan om http requests om te leiden naar https.

```
app.UseHttpsRedirection();
app.UseAuthentication();
app.UseAuthorization();
```

### appsettings.json

In de appsettings.json steken we weer de gegevens van ons Auth0 Domain, onze Auth0 API Identifier, de ClientId en de ClientSecret van onze webapp2.

```
"Auth0": {
  "Domain": "secadvpe-dev.eu.auth0.com",
  "ClientId": "l2V6TJU5S8ZGoRP5M2pka1ex668IAcSq",
  "ClientSecret": "_fEvy27JTjf-bKHxRh-
JYpZw7SGwPUQvc4eGec3yY0rN8P02WbvurTwo00RAMae",
  "Audience": "https://localhost:44339"
}
```

### ConfigureServices

In de *ConfigureServices* methode van de Startup klasse worden de services *AddAuthentication* en *AddJwtBearer* geconfigureerd. In *AddAuthentication* worden de authenticatieschema's ingesteld. Wij gebruiken hier het *CookieAuthenticationDefaults.AuthenticationScheme* van de Nuget package *Microsoft.AspNetCore.Authentication.Cookies*. In de *AddJwtBearer* voeren we de gegevens van onze Authority en onze Audience in. We maken gebruik van de gegevens van het Auth0 Domain en de Auth0 API Identifier uit de appsettings.json. We voegen de *AddCookie* methode toe om authenticatie met behulp van cookies te kunnen doen.

```
services.ConfigureSameSiteNoneCookies();
services.AddAuthentication(options =>
```

```

        {
            options.DefaultAuthenticateScheme =
CookieAuthenticationDefaults.AuthenticationScheme;

            options.DefaultSignInScheme =
CookieAuthenticationDefaults.AuthenticationScheme;

            options.DefaultChallengeScheme =
CookieAuthenticationDefaults.AuthenticationScheme;
        })
        .AddJwtBearer(options =>
        {
            options.Authority = $"https://{Configuration["Auth0:Domain"]}"/;
            options.Audience = $"https://{Configuration["Auth0:Audience"]}"/;
        })
        .AddCookie()

```

Verder maken we gebruik van de OpenID Connect middleware. We configureren de service *AddOpenIdConnect* in *ConfigureServices*.

```

.AddOpenIdConnect("Auth0", options =>
{
    // Set the authority to your Auth0 domain
    options.Authority = $"https://{Configuration["Auth0:Domain"]}";

    // Configure the Auth0 Client ID and Client Secret
    options.ClientId = Configuration["Auth0:ClientId"];
    options.ClientSecret = Configuration["Auth0:ClientSecret"];

    // Set response type to code
    options.ResponseType = OpenIdConnectResponseType.Code;

    // Configure the scope
    options.Scope.Clear();
    options.Scope.Add("openid");
    options.Scope.Add("admin");
    options.Scope.Add("team");
}

```

```

        // Set the callback path

        // Also ensure that you have added the URL as an Allowed Callback URL
in your Auth0 dashboard

        options.CallbackPath = new PathString("/callback");

        // Configure the Claims Issuer to be Auth0
        options.ClaimsIssuer = "Auth0";

        // Saves tokens to the AuthenticationProperties
        options.SaveTokens = true;

        options.Events = new OpenIdConnectEvents
        {
            // handle the logout redirection
            OnRedirectToIdentityProviderForSignOut = (context) =>
            {
                var logoutUri =
$"https://{Configuration["Auth0:Domain"]}/v2/logout?client_id={Configuration["Auth0:C
lientId"]}";

                var postLogoutUri = context.Properties.RedirectUri;
                if (!string.IsNullOrEmpty(postLogoutUri))
                {
                    if (postLogoutUri.StartsWith("/"))
                    {
                        // transform to absolute
                        var request = context.Request;
                        postLogoutUri = request.Scheme + "://" + request.Host
+ request.PathBase + postLogoutUri;
                    }

                    logoutUri += $"&returnTo={
Uri.EscapeDataString(postLogoutUri)}";

```

```

    }

    context.Response.Redirect/logoutUri);
    context.HandleResponse();

    // The context's ProtocolMessage can be used to pass along
additional query parameters
    // to Auth0's /authorize endpoint.
    //
    // Set the audience query parameter to the API identifier to
ensure the returned Access Tokens can be used
    // to call protected endpoints on the corresponding API.
    context.ProtocolMessage.SetParameter("audience",
Configuration["Auth0:Audience"]);

    return Task.CompletedTask;
},
OnRedirectToIdentityProvider = context =>
{
    // The context's ProtocolMessage can be used to pass along
additional query parameters
    // to Auth0's /authorize endpoint.
    //
    // Set the audience query parameter to the API identifier to
ensure the returned Access Tokens can be used
    // to call protected endpoints on the corresponding API.
    context.ProtocolMessage.SetParameter("audience",
Configuration["Auth0:Audience"]);

    return Task.FromResult(0);
}
});
});

```

## AccountController

De AccountController bevat de Login en Logout acties. Logout kan enkel na autorisatie.

```
public async Task Login(string returnUrl = "/")
{
    await HttpContext.ChallengeAsync("Auth0", new AuthenticationProperties()
{ RedirectUri = returnUrl });
}

[Authorize]
public async Task Logout()
{
    await HttpContext.SignOutAsync("Auth0", new AuthenticationProperties
{
    // Indicate here where Auth0 should redirect the user after a logout.
    // Note that the resulting absolute Uri must be whitelisted in the
    // **Allowed Logout URLs** settings for the client.
    RedirectUri = Url.Action("Index", "Home")
});
    await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
}
```

*Geïmplementeerde pagina's:*

<https://localhost:44358/Login>

<https://localhost:44358/Logout>

<https://localhost:44358/Account/Claims>

## StudentsController en PoemController

De StudentsController bevat een aantal routes en gekoppelde acties, waaronder de route /Students met de actie Students.

```
[Route("Students")]
public async Task<IActionResult> Students()
{
```

```

var data = new List<StudentViewModel>();

string accessToken = await HttpContext.GetTokenAsync("access_token");
var client = new RestClient("https://localhost:44339/api/students");
var request = new RestRequest(Method.GET);
request.AddHeader("content-type", "application/json");
request.AddHeader("authorization", "Bearer " + accessToken);
IRestResponse response = client.Execute(request);

if (response.IsSuccessful)
{
    var model = response.Content;
    data = JsonConvert.DeserializeObject<List<StudentViewModel>>(model);
    return View(data);
}
else
{
    throw new Exception("Unable to get content");
}
}

```

*Geïmplementeerde pagina's:*

<https://localhost:44358/Students>

<https://localhost:44358/Students/1>

<https://localhost:44358/Students/1/Edit>

<https://localhost:44358/Students/1/Delete>

<https://localhost:44358/Students/Create>

De PoemController bevat de route /Poem met de actie Poem.

```

public async Task<IActionResult> Poem()
{
    var data = new PoemViewModel();
}

```

```

        string accessToken = await HttpContext.GetTokenAsync("access_token");
        var client = new RestClient("https://localhost:44339/api/poems");
        var request = new RestRequest(Method.GET);
        request.AddHeader("content-type", "application/json");
        request.AddHeader("authorization", "Bearer " + accessToken);
        IRestResponse response = client.Execute(request);

        if (response.IsSuccessfull)
        {
            var model = response.Content;

            List<PoemViewModel> poemsList =
JsonConvert.DeserializeObject<List<PoemViewModel>>(model);

            Random random = new Random();
            data = poemsList[random.Next(poemsList.Count)];
            return View(data);
        }
        else
        {
            throw new Exception("Unable to get content");
        }
    }
}

```

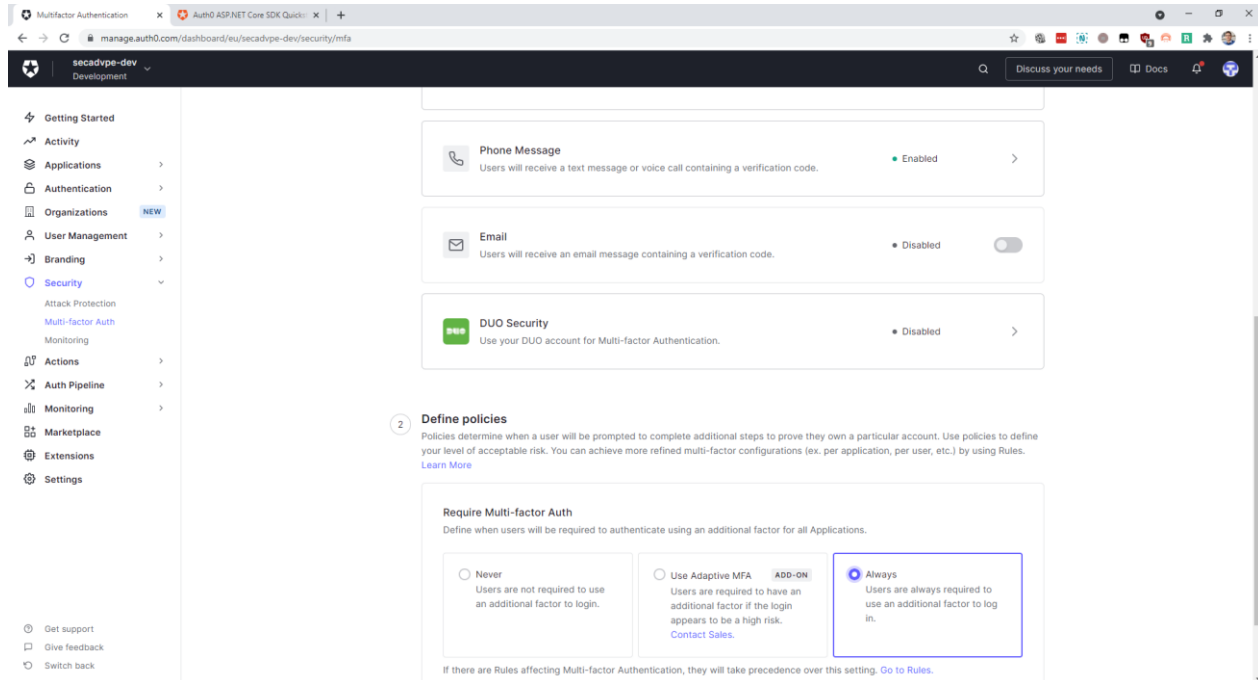
*Geïmplementeerde pagina's:*

<https://localhost:44358/Poem>

Al deze acties doen requests naar onze Web API, bouwen een model op basis van de response en geven dan een view terug van dat model. Dit alles gebeurt op een manier die vergelijkbaar is met de manier van werken in de webapp1. Hier heb ik wel geen gebruik gemaakt van een aparte ApiAccessClient service, maar worden de requests opgebouwd in de controllers zelf. We gebruiken HttpContext.GetTokenAsync() om de tokens op te halen.

## Two-Factor Authentication (Level A+)

2FA via sms werd geïmplementeerd. Dit is eenvoudig in te stellen via de user interface van Auth0.



Bij het registreren moet je een telefoonnummer ingeven en ontvang je een herstelcode.





## Bijna klaar!

Kopieer deze herstelcode en bewaar deze op een veilige plaats. U hebt deze code nodig als u zich ooit zonder uw apparaat moet aanmelden.

ZJFV6P5BP4GH1PVH6NCQ8AP8

Code kopiëren

☐

Ik heb deze code veilig opgenomen

Doorgaan

Wanneer je wil inloggen moet je je identiteit verifiëren door een code die via sms wordt verzonden in te geven.



## Uw identiteit verifiëren

We hebben een sms-bericht verzonden naar:

XXXXXXXX5392

Voer de 6-cijferige code in

|

☐ Dit apparaat 30 dagen onthouden

Doorgaan

Hebt u geen code ontvangen? [Opnieuw verzenden](#)

[Probeer een andere methode](#)

22:14



← 89635



22:12



224760 is your verification code for PXL

1 min • via Mobile Vikings



Bericht sturen (M...





## Uw identiteit verifiëren

We hebben een sms-bericht verzonden naar:

XXXXXXXX5392

Voer de 6-cijferige code in

224760|

☐ Dit apparaat 30 dagen onthouden

Doorgaan

Hebt u geen code ontvangen? [Opnieuw verzenden](#)

[Probeer een andere methode](#)







## Social connections (Level A+)

Login via Twitter en Github werd geïmplementeerd. Google is standaard ingesteld door Auth0 en gebruikt Auth0 development keys (enkel bedoeld voor testing). We hebben ingesteld dat je (enkel) in webapp3 kan inloggen via Github en Twitter.

### Social Connections

[+ Create Connection](#)













Configure social connections like Facebook, Twitter, Github and others so that you can let your users login with them. [Learn more](#) ►

	<b>github</b> GitHub	● 2 Applications enabled	
	<b>google-oauth2</b> ⚠ Google / Gmail	● 5 Applications enabled	
	<b>twitter</b> Twitter	● 2 Applications enabled	

Een nieuwe social connection kan door op 'Create Connection' te klikken en dan een bepaalde connectie te kiezen.

[← Social Connections](#)

## New Social Connection

 <b>Google / Gmail</b> SOCIAL CONNECTION <span>ADDED</span> Allow your users to login with their Google Account	 <b>Facebook</b> SOCIAL CONNECTION A fast and convenient way for users to log into your app with Facebook	 <b>Apple</b> SOCIAL CONNECTION The easy way to add Sign in with Apple to your app or website
 <b>Microsoft Account</b> SOCIAL CONNECTION Enable your users via their trusted Microsoft Account	 <b>LinkedIn</b> SOCIAL CONNECTION Leverage the largest professional social network to enhance your sign-in...	 <b>GitHub</b> SOCIAL CONNECTION Enable the GitHub login option for your Auth0 applications
 <b>Dropbox</b> SOCIAL CONNECTION Auth0 with Dropbox provides social access from the world's leading platform...	 <b>Bitbucket</b> SOCIAL CONNECTION Enable the Bitbucket login option for your Auth0 applications	 <b>PayPal</b> SOCIAL CONNECTION <span>ADDED</span> Allow your users to Sign Up and Log In to your app with PayPal
 <b>PayPal (sandbox)</b> SOCIAL CONNECTION Test your code end-to-end with PayPal Sandbox	 <b>Twitter</b> SOCIAL CONNECTION Twitter allows users to enjoy the benefits of login with as little as one click	 <b>Line</b> SOCIAL CONNECTION Let LINE users easily log in and connect to your app

We moeten dan onze identificatiegegevens bij de social identity provider ingeven. Voor Twitter is dat de Consumer API Key en de Consumer API Secret Key. Voor Github is dat een Client ID en een Client Secret. Om dit te doen moeten we eerst een developer account aanmaken bij de social identity provider. We moeten dan een app aanmaken en verkrijgen dan onze identificatiegegevens.

WEB APP 3 (PE SECURITY ADVANCED YELLOW TEAM)

Web App 3

SettingsKeys and tokens

☰


App Details

Edit

NAME

Web App 3

APP ICON



APP ID

20946943

DESCRIPTION

This app was created to use the Twitter API.  
This information will be visible to people who've authorized your app

↔

App permissions

Edit

Read Only


Read Tweets and profile information

✓

Authentication settings

Edit

3-legged OAuth is enabled



Use 3-legged OAuth for Sign in with Twitter, posting Tweets on behalf of other accounts and more. Get more information in the [docs](#).

CALLBACK URLS

<https://secadvpe-dev.eu.auth0.com/login/callback>

WEBSITE URL

<https://secadvpe-dev.eu.auth0.com>

## Web App 3

Settings **Keys and tokens**

## Consumer Keys ⓘ

API Key and Secret

Regenerate

## Authentication Tokens ⓘ

Bearer Token

Generated May 22, 2021

Regenerate

Revoke

Access Token and Secret

Generated May 22, 2021

For @janvinkenrooye

Created with [Read Only](#) permissions

Regenerate

Revoke



We kunnen deze dan ingeven in Auth0 en onze social connection verder instellen (attributes en permissions aanduiden).

← Social Connections



**twitter**

Twitter Identifier `con_70BHzJkJMt52io8o`

▶ Try Connection

📖 Setup guide

Settings

Applications

### General

#### Name

twitter

If you are triggering a login manually, this is the identifier you would use on the connection parameter.

#### Consumer API key

gjEhvw7qUmMINC0O7Yap79PAQ

[How to obtain a Consumer API key?](#)

#### Consumer API secret key

.....



For security purposes, we don't show your existing Consumer API secret key.

#### Attributes

☒ Basic Profile **REQUIRED** ⓘ

### Advanced

Sync user profile attributes at each login



Save changes

We moeten dan nog de applicaties kiezen die toegang hebben tot deze sociale connectie.

← Social Connections



**twitter**

Twitter Identifier `con_708HzJkJMt52io8o`

▶ Try Connection

📖 Setup guide

Settings Applications

Applications using this connection.



Auth0 Management API (Test Application)  
Machine to Machine



Postman  
Machine to Machine



Web App 1  
Machine to Machine



Web App 2  
Regular Web Application



Web App 3  
Regular Web Application



Het resultaat is dat we ons kunnen registreren via onze sociale connecties.

## Geef je Web App 3 toegang tot je account?

☐ Ingelogd blijven · [Wachtwoord vergeten?](#)

**Deze applicatie kan:**

- Tweets van je tijdlijn bekijken (waaronder afgeschermdde Tweets), en je lijsten en collecties.
- Je profielgegevens en accountinstellingen van Twitter bekijken.
- Accounts bekijken die je volgt, negeert en blokkeert.

Ga voor meer informatie over toestemmingen van apps van derden naar ons [Helpcentrum](#).




**Web App 3**  
secadvpe-dev.eu.auth0.com  
This app was created to use the Twitter API.

We raden je aan de voorwaarden en het privacybeleid van de app te bekijken om te zien hoe deze gebruikmaakt van gegevens van je Twitter-account. Je kan toegang voor elke app altijd intrekken via [Apps en sessies](#) bij je accountinstellingen van Twitter.

Door toestemming te verlenen aan een app, geef je aan akkoord te gaan met de [algemene voorwaarden van Twitter](#). Met name sommige gebruiksinformatie wordt gedeeld met Twitter. Bekijk voor meer informatie ons [privacybeleid](#).


43

We kunnen ook inloggen via onze sociale connecties.



## Welkom

Meld u aan bij PXL om door te gaan naar Web App 3.





[Wachtwoord vergeten?](#)


Doorgaan

Hebt u geen account? [Aanmelden](#)

OF

 Ga verder met Google

 Ga verder met GitHub

 Ga verder met Twitter