**Section09 Securing the API**

**Notes: -**

**1-in the previous section we learn about how to store JWT token as cookie on the Client project by make callback URL on the Authorized action to redirect to target action authorize on controller o auth and then when click submit it will create cookie and resend again to the client and the stored cookie is readable for the both sides.**

**(but you cannot control the validate of the returned token from the server side)**

**Steps: -**

**1-create new Empty .net core project called Api**

**2-on the startup.cs we set the following code as below**

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Builder;**

**using Microsoft.AspNetCore.Hosting;**

**using Microsoft.AspNetCore.Http;**

**using Microsoft.Extensions.DependencyInjection;**

**using Microsoft.Extensions.Hosting;**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Security.Claims;**

**using System.Threading.Tasks;**

**namespace Api{**

**public class Startup{**

**public void ConfigureServices(IServiceCollection services){**

**services.AddAuthentication();**

**services.AddAuthorization(config =>{**

**var defaultAuthBuilder = new AuthorizationPolicyBuilder();**

**var defaultAuthPolicy = defaultAuthBuilder**

**.AddRequirements(new JwtRequirement())**

**.Build();**

**config.DefaultPolicy = defaultAuthPolicy;});**

**services.AddScoped<IAuthorizationHandler, JwtRequirementHandler>();**

**//to allow inject HttpClient to send the request**

**//to allow inject HttpContextAccessor to recieve the request**

**services.AddHttpClient().AddHttpContextAccessor();**

**services.AddControllers();}**

**// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.**

**public void Configure(IApplicationBuilder app, IWebHostEnvironment env){**

**if (env.IsDevelopment()){app.UseDeveloperExceptionPage();}**

**app.UseRouting();**

**app.UseAuthentication();**

**app.UseAuthorization();**

**app.UseEndpoints(endpoints =>{endpoints.MapDefaultControllerRoute();});}}}**

**3-create folder called JwtRequirement and then create class JwtRequirement.cs**

**//the requirement handler will hit for each request coming from the client app with the access\_token which contain the JWT token and then make request to the server app with the OAuth > validate action**

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Http;**

**using System.Net;**

**using System.Net.Http;**

**using System.Threading.Tasks;**

**namespace Api{**

**public class JwtRequirement : IAuthorizationRequirement{}**

**public class JwtRequirementHandler : AuthorizationHandler<JwtRequirement>{**

**private readonly HttpClient \_client;**

**private readonly HttpContext \_httpContext;**

**public JwtRequirementHandler(IHttpClientFactory httpClientFactory,IHttpContextAccessor httpContext){**

**\_client = httpClientFactory.CreateClient();**

**\_httpContext = httpContext.HttpContext;}**

**protected override async Task HandleRequirementAsync(**

**AuthorizationHandlerContext context,**

**JwtRequirement requirement){**

**if (\_httpContext.Request.Headers.TryGetValue("Authorization", out var authHeader)){**

**var accessToken = authHeader.ToString().Split(' ')[1];**

**var response = await \_client.GetAsync**

**($"https://localhost:44382/oauth/validate?access\_token={accessToken}");**

**if (response.StatusCode == HttpStatusCode.OK){context.Succeed(requirement);}}}}}**

**4-on the server > OAuth > create new action called validate and set the following code**

**[Authorize]**

**public IActionResult Validate(){**

**if (HttpContext.Request.Query.TryGetValue("access\_token", out var accessToken))**

**{return Ok();}**

**return BadRequest();}**

**5-on the client > HomeController > Secret action we update by the following code**

**[Authorize]**

**public async Task<IActionResult> Secret(){**

**var token = await HttpContext.GetTokenAsync("access\_token");**

**\_client.DefaultRequestHeaders.Add("Authorization", $"Bearer {token}");**

**var serverResponse = await \_client.GetAsync("https://localhost:44382/Secret/Index");**

**var apiResponse = await \_client.GetAsync("https://localhost:44345/Secret/Index");**

**return View();}**

**so, the operation is happening as below**

**when make request to the client API > Home >Secret it will redirect to the Server > OAuth > Authorize to generate the JWT token and store it on the client and return to the Home > secret to make request to the Api that redirect to the authorization handler that make validate access token by send another request to the server > OAuth > Validate and then return if its validated or not**