WorkshopPLUS – Modern Authentication and Authorization

Developing applications

Student Lab Manual

Instructor Edition (Book Title Hidden Style)

Version 2.1

**Conditions and Terms of Use**

**Microsoft Confidential - For Internal Use Only**

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2020 Microsoft Corporation. All rights reserved.

**Copyright and Trademarks**

© 2016 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at  
[*http*://www.microsoft.com/about/legal/permissions/](http://www.microsoft.com/about/legal/permissions/)

Microsoft®, Internet Explorer®, and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Contents

[Introduction: 6](#_Toc508614491)

[Exercise 1 - Create a Web API application 7](#_Toc508614492)

[Exercise 2 – Create a Web UI application 8](#_Toc508614493)

[Exercise 3 – Add authorization to the Web API project 10](#_Toc508614494)

[Exercise 4 – Add access token request to the WebUI project 10](#_Toc508614495)

[Exercise 5 – Add application roles 12](#_Toc508614496)

# Introduction:

#### Objectives

The purpose of this lab is to help the student acquire hands on experience of developing an application using the common identity infrastructure. By the end of the lab the student will have developed an application consisting of the following components and functionality:

1. A Web UI application serving html pages, able to authenticate the user.
2. The Web UI application will use role-based access control.
3. A Web API application providing a REST interface used by the Web UI application. Access to the APIs will be restricted to authorized users.

#### Prerequisites (if applicable)

VS.NET 2019

Web browser

#### Estimated time to complete this lab

2.0h

# Exercise 1: Create a Web API application

|  |  |  |
| --- | --- | --- |
|  | **Procedure** | **Purpose** |
| 1.0 | In VS.NET 2019 create a new, empty solution. Call it AADLab. (In Create Project dialog search for ’empty solution’) | Create solution for our projects |
| 1.1 | Add a new C# ASP.NET API project (ASP.NET Core Web Application). Name project WebAPI. Click **Create**. | Initiate creation of a simple Web API application. |
| 1.2 | Select API template and press **Create** | Create a Web API project with no access control |
| 1.3 | In the Azure AAD portal blade use *App Registration* option to create a new application with display name WebAPI. Use the *Expose API* option to set the Application ID Uri to *api://webapi* and add one scope: *scope1* | Register the API in your AAD tenant |
| 1.4 | Modify the appSettings.json file to include the following section:  *"AzureAd": {*  *"Instance": "https://login.microsoftonline.com/",*  *"Domain": "<your domain>.onmicrosoft.com",*  *"TenantId": "<tenant id>",*  *"ClientId": "<application id>",*  *"Audience": "https://webapi"*  *}* | Provide AAD configuration data to your application |
| 1.5 | Use NuGet manager to add reference to Microsoft.Identity.Web | Add support for token validation functionality |
| 1.6 | Add the as first line of code in the ConfigureServices method (Startup class):  services.AddMicrosoftIdentityWebApiAuthentication(Configuration); | Inject token validation functionality into the http pipeline |

# Exercise 2 – Create a Web UI application

|  |  |  |
| --- | --- | --- |
|  | Procedure | Purpose |
| 2.1 | Add a new C# ASP.NET UI project (ASP.NET Core Web Application. Name project WebUI. Press **Create.** Select the Model-View-Controller template. Press Create | Initiate creation of a simple Web UI application. |
| 2.2 | In the Azure AAD portal blade use *App Registration* option to create a new application with display name WebUI with reply url of your newly created application. Use the *API* Permissions option to register use of the WebAPI scope1 created in the previous exercise. Add a new secret to the application. | Configure use of the OpenIDConnect/OAuth2 protocol in AAD |
| 2.3 | Modify the appsettings.json file to include the following section. Replace <> values with values from your application registration.   |  | | --- | | "AzureAd": {  "Instance": "https://login.microsoftonline.com/",  "Domain": "<tenant name>.onmicrosoft.com",  "TenantId": "<tenant id>",  "ClientId": "<application id>",  "CallbackPath": "/signin-oidc",  "RedirectUri": "https://localhost:<port>/signin-oidc",  "AadAuthorityAudience": "AzureAdMyOrg",  "EnablePiiLogging": true  } | | Use project context menu to open the secrets.json file and add the application secret created in 2.2 above as follows:   |  | | --- | | "AzureAd:ClientSecret": "<secret>" | |  | | | Provide configuration data to your application |
| 2.4 | Open the HomeController in the Web UI application and replace its Privacy method with:   |  | | --- | | public async Task<IActionResult> Privacy()  {  var client = new HttpClient();  var res = await client  .GetStringAsync("https://localhost:  <webapi port no>  /api/WeatherForecast");  return View();  } |   Replace the <webapi port no> with the port number of your WebAPI project (see it’s sslPort in launchSettings.json file). | Call Web API from the UI app |
| 2.5 | Right-click on the solution in the Solution Explorer and select Properties. Select Multiple Startup Project in the dialog and choose Start option for both projects. | Set both projects to run in debug simultaneously |
| 2.6 | Put a break on the return View() line. Press F5 and click on the Privacy option when the browser shows your page. Make sure you received a valid response. | Test initial solution, calling the API without a token. |

# Exercise 3 – Add authorization to the Web API project

|  |  |  |
| --- | --- | --- |
|  | Procedure | Purpose |
| 3.1 | Add the [Authorize] attribute in the WebAPI project’s WeatherForecastController before the class definition line.  Run your solution and select the Privacy option. You should get an Unauthorized response. | Change API to require an OAuth2 authorization token. |

# Exercise 4 – Add access token request to the WebUI project

|  |  |  |
| --- | --- | --- |
| 4.3 | Use nugget manager to add *Microsoft.Identity.Web* to the WebUI project | Add OAuth2 token acquisition toolkits |
| 4.4 | Replace Startup.ConfigureServices with the following code (from <https://github.com/Azure-Samples/active-directory-aspnetcore-webapp-openidconnect-v2/tree/master/2-WebApp-graph-user/2-1-Call-MSGraph>)   |  | | --- | | public static string[] Scopes = { "https://webapi/scope1" };  public void ConfigureServices(IServiceCollection services)  {  services.Configure<CookiePolicyOptions>(options =>  {  // This lambda determines whether user consent for non-essential cookies is needed for a given request.  options.CheckConsentNeeded = context => true;  options.MinimumSameSitePolicy = SameSiteMode.Unspecified;  // Handling SameSite cookie according to https://docs.microsoft.com/en-us/aspnet/core/security/samesite?view=aspnetcore-3.1  options.HandleSameSiteCookieCompatibility();  });  services.AddOptions();  services.AddMicrosoftIdentityWebAppAuthentication(Configuration)  .EnableTokenAcquisitionToCallDownstreamApi(Scopes)  .AddInMemoryTokenCaches();  services.AddControllersWithViews(options =>  {  var policy = new AuthorizationPolicyBuilder()  .RequireAuthenticatedUser()  .Build();  options.Filters.Add(new AuthorizeFilter(policy));  }).AddMicrosoftIdentityUI();  services.AddRazorPages();  } | | Performs authentication using Authorization Code grant, exchanges the code for an access token to WebAPI |
| 4.3 | Modify the constructor for the HomeController with the following:   |  | | --- | | ITokenAcquisition \_tokenAcquisition;  public HomeController(ITokenAcquisition tokenAcquisition)  {  \_tokenAcquisition = tokenAcquisition;  } | | Injects token acquisition functionality into the controller |
| 4.4 | Insert the following lines into the Privacy method after the line creating the HttpClient:   |  | | --- | | var token = await \_tokenAcquisition.GetAccessTokenForUserAsync(Startup.Scopes);  client.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", token); | | Get current access token to WebAPI, from cache or by using the refresh token. |
| 4.5 | Run both project and verify the API is being called. | Test |