Integration with Launch Pad

Michigan

# Overview

The Launch Pad is an application that serves as a landing page for a user that has logged into the midatahub.org SSO. After successfully logging in, it will display all applications within the SSO ecosystem to which the authenticated user has access. To check for access, the Launch Pad pings each application in the midatahub.org SSO that requires security to determine if the user has authorizations for the application. If user authorization is found, the application will appear on the user’s Launch Pad page.

# SSO App Implementation

To ping an application in the midatahub.org SSO, the Launch Pad encrypts the username of a user using JSON Web Token (JWT) and signs it. The resulting application then decodes the JWT Token and determines whether or not the user has access to their application. If they do, the application will return a 200 (OK), otherwise, a 404 (NOT FOUND).

The example below assumes that Web Api is used.

1. **Create RsaKeyUtils**

This code block is used to import the RsaPublicKey file. This will be used to validate the IssuerSigningCredentials.

Code Block 1 RSA Key Utils

|  |
| --- |
| using System.IO;using System.Security.Cryptography;using Newtonsoft.Json;namespace RsaUtilsNamespace{ public class RsaKeyUtils { public static RSAParameters GetKeyParameters(string file) { if (!File.Exists(file))  throw new FileNotFoundException(string.Format("Check configuration - cannot find auth key file: {0}", file)); var keyParams = JsonConvert.DeserializeObject<RsaParametersWithPrivate>(File.ReadAllText(file)); return keyParams.ToRsaParameters(); } /// <summary> /// Util class to allow restoring RSA parameters from JSON as the normal /// RSA parameters class won't restore private key info. /// </summary> private class RsaParametersWithPrivate { public byte[] D { get; set; } public byte[] DP { get; set; } public byte[] DQ { get; set; } public byte[] Exponent { get; set; } public byte[] InverseQ { get; set; } public byte[] Modulus { get; set; } public byte[] P { get; set; } public byte[] Q { get; set; } public RSAParameters ToRsaParameters() { return new RSAParameters { D = D, DP = DP, DQ = DQ, Exponent = Exponent, InverseQ = InverseQ, Modulus = Modulus, P = P, Q = Q }; } } }} |

1. **Create the JwtTokenProvider**

This is the token provider to decrypt the JWT Token. The issuer from the Launch Pad will always be "Launch Pad." Replace the audience string with your own application name or custom audience. Next, share the audience value with the Launch Pad administrator to be set in the Launch Pad. The "RsaPublicKey" refers to the file name in the AppSettings of the public key file. The following example assumes that the public key is at the root of the project.

Code Block 2 JWT Token Provider

|  |
| --- |
| using System;using System.Configuration;using System.IdentityModel.Tokens;using System.IO;using System.Security.Claims;using System.Security.Cryptography;using RsaUtilsNamespace;namespace Namespace{ public interface IJwtTokenProvider { ClaimsPrincipal DecryptToken(string tokenString); } public class JwtTokenProvider : IJwtTokenProvider { public ClaimsPrincipal DecryptToken(string tokenString) { var audience = "Insert Name of Application Here"; var validationParameters = new TokenValidationParameters { ValidIssuer = "Launch Pad", ValidAudience = audience, IssuerSigningKey = GetRsaSecurityKey(), ValidateAudience = true, ValidateIssuerSigningKey = true }; var handler = new JwtSecurityTokenHandler(); SecurityToken securityToken; var principal = handler.ValidateToken(tokenString, validationParameters, out securityToken); return principal; } private static RsaSecurityKey GetRsaSecurityKey() { var directory = AppDomain.CurrentDomain.BaseDirectory; var rsaKeyFile = ConfigurationManager.AppSettings["RsaPublicKey"]; var keyParameters = RsaKeyUtils.GetKeyParameters(Path.Combine(directory, rsaKeyFile)); const int dwKeySize = 2048; var provider = new RSACryptoServiceProvider(dwKeySize); provider.ImportParameters(keyParameters); var key = new RsaSecurityKey(provider); return key; } }} |

Code Block 3 Public Key

|  |
| --- |
| { "Exponent": "AQAB", "Modulus": "3BICwqptrUCrWfejb/10afQsQ07+l7wvWAKeDWag2PIDgb+/uLyHzMNMZ4l8gFX7Cqfs9G0nFAVkpCzxYgWQvL5Lr5LI5UNg54/QWbsf9HSffWyZGMhpiXmJDW15O8LCezZZMEzJuzk9AOABwwNwQ5Bs8UeXPSsm6r1C5zX8ZONCagXxrN2gEuMyDZ9BmCvITEkVY8R1s9TWT/DwJvXnbgG/7627ePGDW2CpCnIEA8aRLfKYP8xJ03uT4mBsXKDdY/WCellYZOCFKOt+K0+BOoU7SVHwy4zs0bzOb9I/j4pcUjvhny6wgqUQIWV5S5VkvUi15hVmNfgFu/hKzx3Ktw=="} |

1. **Create the Access Service**
The sample below uses the created IJwtTokenProvider to decrypt the incoming token from the Launch Pad. The only claim is "username". Once retrieved, use the username claim to determine whether or not a user has access to the application.

Code Block 4 Access Service

|  |
| --- |
| namespace ServiceNamespace{ public interface IAccessService { bool Get(string token); } public class AccessService : IAccessService { private readonly IJwtTokenProvider \_jwtTokenProvider; public AccessService(IJwtTokenProvider jwtTokenProvider) { \_jwtTokenProvider = jwtTokenProvider; } public bool Get(string token) { var decrypted = \_jwtTokenProvider.DecryptToken(token); var username = decrypted.Claims.FirstOrDefault(x => x.Type == "username"); if (username == null) return false; // Implement Code To Determine if User has access } }} |

1. **Create Access Api** Controller
The Launch Pad passes the api in the query string as "token". Example Link: https://mywebsite.com/api/access?token=178uhdsf98ahrp23n09svsnfs. The api will need to add the cross origin in order to be successfully accessed by the Launch Pad.

Code Block 5 Access Controller

|  |
| --- |
| using System.Net;using System.Net.Http;using System.Web.Http;namespace OontrollerNamespace{ [RoutePrefix("api")] public class AccessController : ApiController { private readonly IAccessService \_accessService; public AccessController(IAccessService accessService) { \_accessService = accessService; } [HttpGet] [Route("access")] [AllowAnonymous] public HttpResponseMessage HasAccess(string token) { var hasAccess = \_accessService.Get(token); var message = Request.CreateResponse(hasAccess ? HttpStatusCode.OK : HttpStatusCode.NotFound); message.Headers.Add("Access-Control-Allow-Origin", "\*"); return message; } }} |

# Launch Pad Implementation

To add an application to the Launch Pad, an administrator user will need to add the audience, api url and url.

