AWS Web Identity Federation for Mobile Apps - Facebook (1 of 3 series)

Sivaprasad Padisetty (February 2014)

*Imagine that you have a mobile app that needs access to AWS resources. (Or it might be a web app that uses client script; the concepts presented here are the same.) The app might be a game that runs on a phone and stores the player and score information in an Amazon S3 bucket or an Amazon DynamoDB table. Because the app needs to be able to distinguish individual users, users cannot be anonymous.*

*Since it is not a good idea to distribute long-term credentials, you want to build the app such that it requests temporary security credentials using web identity federation. This lets you create an app that authenticates users using existing identity providers like a) Facebook, b) Login with Amazon, c) Google.*

*Using any of these providers can simplify the development and management of your app. Instead of providing custom sign-in logic and having to manage user login information (either in a custom system or as IAM users), your app can rely on well-known and secure sign-in protocols that many users already have access to. Because you can trade a token from the identity provider for temporary security credentials, you don't have to distribute any credentials with the app, and you don't need to manage the process of rotating the credentials. You can find more info* [*here*](http://docs.aws.amazon.com/STS/latest/UsingSTS/CreatingWIF.html)*.*

*This is a three part series, where each part will cover one identity provider. I will cover Facebook in this blog.*

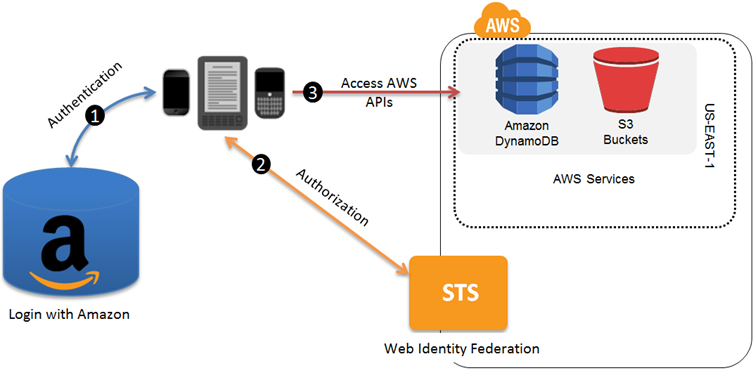
# 

# Prerequisites and Helper function

* Sign up for AWS and get the AccessKey & SecretKey. You can find the info about AWS Account and Access Keys [here](http://docs.aws.amazon.com/powershell/latest/userguide/pstools-appendix-signup.html).
* You have Visual Studio installed, I used Visual Studio 2013. Although I did not test it, earlier version should work.
* Install AWS SDK for .Net from [here](http://aws.amazon.com/sdkfornet/). Follow the “Getting Started” instructions from there.

# Mobile App with Third Party Sign-In

The following figure shows a simplified flow for how this might work, using Login with Amazon as the identity provider. For Step 1, the app can also invoke Facebook or Google, but that's not shown here.



The following details enable this scenario:

* Developer has registered the mobile app with different identity providers, who have assigned an app ID to the app.
* The mobile app includes logic to invoke the appropriate identity provider (depending on which sign-in option the user chooses) and to get back a token from the provider.
* The app can call AssumeRoleWithWebIdentity without using any AWS security credentials. The call includes the token received from the provider previously.
* AWS STS is able to verify that the token passed from the app is valid and then returns temporary security credentials to the app. The mobile app's permissions to access AWS are established by the role that the app assumes.

# Create a Role

This is the role that a Facebook authenticated user will assume. The role is associated with two things a) trust policy – who can assume this role and b) access policy – what permission does the assumed user have.

C# code below creates a role. Normally, this is manually created once. I chose to write C# code because it is handy for automation. This role can be assumed by any authenticated Facebook user. The user only has access to their specific key which is located under “federationbucket/Facebook/<userid>”. Code below is slightly complicated because the same code is used for Google/Amazon login as well.

identityProvider = "Facebook";

providerURL = "graph.facebook.com";

providerAppIdName = "app\_id";

providerUserIdName = "id";

//identity provider specific AppId is loaded from app.config

// The key names are like FacebookProviderAppId. GoogleProviderAppId, AmazonProviderAppId

providerAppId = ConfigurationManager.AppSettings[identityProvider + "ProviderAppId"];

// Since the string is passed String.Format, '{' and '}' has to be escaped.

// Policy document specifies who can invoke AssumeRoleWithWebIdentity

string trustPolicyTemplate = @"{{

""Version"": ""2012-10-17"",

""Statement"": [

{{

""Effect"": ""Allow"",

""Principal"": {{ ""Federated"": ""{1}"" }},

""Action"": ""sts:AssumeRoleWithWebIdentity"",

""Condition"": {{

""StringEquals"": {{""{1}:{2}"": ""{3}""}}

}}

}}

]

}}";

// Defines what permissions to grant when AssumeRoleWithWebIdentity is called

string accessPolicyTemplate = @"{{

""Version"": ""2012-10-17"",

""Statement"": [

{{

""Effect"":""Allow"",

""Action"":[""s3:GetObject"", ""s3:PutObject"", ""s3:DeleteObject""],

""Resource"": [

""arn:aws:s3:::federationtestbucket/{0}/${{{1}:{4}}}"",

""arn:aws:s3:::federationtestbucket/{0}/${{{1}:{4}}}/\*""

]

}}

]

}}";

CreateRoleRequest createRoleRequest = new CreateRoleRequest

{

RoleName = "federationtestrole",

AssumeRolePolicyDocument = string.Format(trustPolicyTemplate, identityProvider, providerURL, providerAppIdName, providerAppId)

};

Console.WriteLine("\nTrust Policy Document:\n{0}\n", createRoleRequest.AssumeRolePolicyDocument);

CreateRoleResponse createRoleResponse = iamClient.CreateRole(createRoleRequest);

PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest

{

PolicyName = "federationtestrole-rolepolicy",

RoleName = "federationtestrole",

PolicyDocument = string.Format(accessPolicyTemplate, identityProvider, providerURL, providerAppIdName, providerAppId, providerUserIdName)

};

Console.WriteLine("\nAccess Policy Document (Permissions):\n{0}\n", putRolePolicyRequest.PolicyDocument);

PutRolePolicyResponse putRolePolicyResponse = iamClient.PutRolePolicy(putRolePolicyRequest);

System.Threading.Thread.Sleep(5000);

AmazonS3Config config = new AmazonS3Config

{

ServiceURL = "s3.amazonaws.com",

RegionEndpoint = Amazon.RegionEndpoint.USEast1

};

Above code assumes an app.config file to contain the following values.

<appSettings>

<add key="AWSAccessKey" value="YOUR\_ACCESS\_KEY\_A134" />

<add key="AWSSecretKey" value="YOUR\_SECRET\_KEY\_HERE\_SECRET\_KEY\_HEREndgN" />

<add key="AWSRegion" value="us-east-1" />

<add key="FacebookProviderAppId" value="123456789012345" />

<add key="GoogleProviderAppId" value="123456789012-your\_app\_id\_here\_your\_app\_id\_abc.apps.googleusercontent.com" />

<add key="GoogleProviderAppIdSecret" value="your\_app\_secret\_here\_abc" />

<add key="AmazonProviderAppId" value="amzn1.application.your\_app\_id\_here\_your\_appid\_here" />

<add key="AmazonProviderClientId" value="amzn1.application-oa2-client.your\_client\_id\_here\_client\_id\_ab" />

<add key="ClientSettingsProvider.ServiceUri" value="" />

</appSettings>

Trust Policy document produced by the above code:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": { "Federated": "graph.facebook.com" },

"Action": "sts:AssumeRoleWithWebIdentity",

"Condition": {

"StringEquals": {"graph.facebook.com:app\_id": "123456789012345"}

}

}

]

}

Access Policy document (permissions) produced by the above code:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect":"Allow",

"Action":["s3:GetObject", "s3:PutObject", "s3:DeleteObject"],

"Resource": [

"arn:aws:s3:::federationtestbucket/Facebook/${graph.facebook.com:id}",

"arn:aws:s3:::federationtestbucket/Facebook/${graph.facebook.com:id}/\*"

]

}

]

}

# Authenticate with Facebook and get the token

The following query is constructed, and a GET action is performed in the browser. It will be redirected to a page where Facebook gets user password/consent. If the authentication succeeds, it will be redirected to the URL specified in the query. The C# code below uses the Forms based WebBrowser control to automate this process.

The query:

string query = "https://www.facebook.com/dialog/oauth?" +

string.Format ("client\_id={0}&", client\_id) +

"response\_type=token&" +

"redirect\_uri=https://www.facebook.com/connect/login\_success.html";

The GetToken helper function, does GET operation and retrieves the token from the redirected URL.

class MyWebBrowser : WebBrowser

{

public string CapturedUrl;

string token;

public MyWebBrowser(string token)

{

this.token = token + "=";

}

protected override void OnDocumentCompleted(WebBrowserDocumentCompletedEventArgs e)

{

base.OnDocumentCompleted(e);

string st = e.Url.ToString();

if (st.Contains(token))

{

this.Navigate("about:blank"); // hack, closing the form here does not work always.

this.CapturedUrl = st;

Console.WriteLine("Captured: {0}", st);

}

else if (st == "about:blank")

{

((Form)this.Parent).Close();

}

}

}

string GetToken(string token, string url)

{

Form f = new Form();

MyWebBrowser wb = new MyWebBrowser(token);

wb.Dock = DockStyle.Fill;

f.Controls.Add(wb);

wb.Navigate(url);

f.WindowState = FormWindowState.Maximized;

f.ShowDialog();

string st = wb.CapturedUrl;

f.Dispose();

if (st == null)

throw new Exception("Oops! Error getting the token");

int index = st.IndexOfAny(new char[] { '?', '#' });

st = index < 0 ? "" : st.Substring(index + 1);

NameValueCollection pairs = HttpUtility.ParseQueryString(st);

string tokenValue = pairs[token];

Console.WriteLine("TOKEN={0}, Value={1}", token, tokenValue);

return tokenValue;

}

# Get Temporary Credentials with AssumeRoleWithWebIdentity

Key concept to grasp here is, you start with anonymous AWS credentials, pass the Facebook token and get the temporary credentials. This is important because the mobile app user will not have any AWS credentials.

// role - ARN for the role to assume

// client\_id of the Facebook app

public AssumeRoleWithWebIdentityResponse GetTemporaryCredentialUsingFacebook(string client\_id, string role)

{

string query = "https://www.facebook.com/dialog/oauth?" +

string.Format ("client\_id={0}&", client\_id) +

"response\_type=token&" +

"redirect\_uri=https://www.facebook.com/connect/login\_success.html";

AssumeRoleWithWebIdentityRequest assumeRoleWithWebIdentityRequest = new AssumeRoleWithWebIdentityRequest

{

ProviderId = "graph.facebook.com",

WebIdentityToken = GetToken("access\_token", query),

RoleArn = role,

};

return GetAssumeRoleWithWebIdentityResponse (assumeRoleWithWebIdentityRequest);

}

AssumeRoleWithWebIdentityResponse GetAssumeRoleWithWebIdentityResponse(AssumeRoleWithWebIdentityRequest assumeRoleWithWebIdentityRequest)

{

// Start with Anonymous AWS Credentials and get temporary credentials.

var stsClient = new AmazonSecurityTokenServiceClient(new AnonymousAWSCredentials());

assumeRoleWithWebIdentityRequest.DurationSeconds = 3600;

assumeRoleWithWebIdentityRequest.RoleSessionName = "MySession";

return stsClient.AssumeRoleWithWebIdentity(assumeRoleWithWebIdentityRequest);

}

# Using temporary credentials to make an update to S3 bucket

S3Test s3Test = new S3Test();

s3Test.CreateS3Bucket("federationtestbucket",

identityProvider + "/" + assumeRoleWithWebIdentityResponse.SubjectFromWebIdentityToken,

assumeRoleWithWebIdentityResponse.Credentials, config);

class S3Test

{

public void CreateS3Bucket(string bucketName, string key, Credentials credentials, AmazonS3Config config)

{

var s3Client = new AmazonS3Client(credentials.AccessKeyId, credentials.SecretAccessKey, credentials.SessionToken, config);

string content = "Hello World2!";

// Put an object in the user's "folder".

s3Client.PutObject(new PutObjectRequest

{

BucketName = bucketName,

Key = key,

ContentBody = content

});

Console.WriteLine("Updated key={0} with content={1}", key, content);

}

}

# Delete the role created

Again, you will not be doing the following in your mobile app. This is only for automation. Also, note that you need real AWS credentials to create and delete the role.

DeleteRolePolicyResponse deleteRolePolicyResponse = iamClient.DeleteRolePolicy(new DeleteRolePolicyRequest

{

PolicyName = "federationtestrole-rolepolicy",

RoleName = "federationtestrole"

});

DeleteRoleResponse deleteRoleResponse = iamClient.DeleteRole(new DeleteRoleRequest

{

RoleName = "federationtestrole"

});

# Registering an app with Facebook

Create an app at <https://developers.facebook.com/> by providing a name and category. Once the app is created, note down the 15 digit App Id, we will need it later.

Facebook exposed a nice [way](https://developers.facebook.com/tools/explorer) to manually get the token, very handy during development. Also, find the Web Identity Federation link from the references, this is very helpful as you develop/debug your app.

# References

1. [Scenarios for Granting Temporary Access](http://docs.aws.amazon.com/STS/latest/UsingSTS/STSUseCases.html)
2. [Creating Temporary Security Credentials](http://docs.aws.amazon.com/STS/latest/UsingSTS/CreatingWIF.html)
3. [Sample Identity provider blog](http://blogs.aws.amazon.com/net/post/Tx2KW5KYMRE681I/Web-Identity-Federation-using-the-AWS-SDK-for-NET)
4. [Web Identity Federation Playground](https://web-identity-federation-playground.s3.amazonaws.com/index.html)
5. [Facebook login workflow](https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow/)

You can find the code under “AWS\AWS CSharp Test” folder at <https://github.com/padisetty/Samples>.

Explore & Enjoy!

/Siva