Using Humana Application Security Blocks V2.1

Purpose

The purpose of this document is to detail the installation, use and configuration of the Security Application Block

Overview

Humana.ApplicationBlocks.Security dll is an http module that is configured with the application. It intercepts incoming http request to the application, performs authentication and authorization check on the request, and decides if allow the access to the resource, or deny it.

Implementation

The SecurityApplication block consists of three main components: the SecurityModule, the SecurityPrincipal, and the SecurityIdentity.

The SecurityModule is responsible for intercepting incoming requests to an application. It implements the Intercepting Filter design pattern. Requests from authenticated and authorized users are allowed to pass through to the application. Requests that are not from authenticated and authorized users are rejected.

The SecurityPrincipal represents the user. The SecurityModule uses the SecurityPrincipal to determine if a user is authenticated and authorized.

The SecurityIdentity is the identity of the SecurityPrincipal. The SecurityIdentity contains the actual nuts and bolts of authentication and authorization. The SecurityIdentity contains the attributes of the user such as AKA_Name, first name, last name, address, city, AIN to name a few. It is the entity that interacts directly with SecuredLogons to authenticate and authorize.

Configuring the Security Application Block

The majority of the work in using the Security Application Block is configuring it. All of the configuration items reside in the web.config, which is typical for .Net applications. The following are lines you will need to add to your web.config:

In the configSections add:

```
<configSections>

        <section name="humana.security"
        type="Humana.ApplicationBlocks.Security.Config.SecurityConfigHandler,
        Humana.ApplicationBlocks.Security"/>

</configSections>
```

In the system.web section add:

```
    <httpModules>
      <add type="Humana.ApplicationBlocks.Security.HttpModules.SecurityModule,
Humana.ApplicationBlocks.Security" name="SecuredLogonsSecurityModule"/>
    </httpModules>
```

Add a new section to the web config that looks like this:

```
<humana.security
       identityType="Humana.ApplicationBlocks.Security.SecurityIdentity,
       Humana.ApplicationBlocks.Security"
       identityConfigType="Humana.ApplicationBlocks.Security.Config.SecuredLogonsI
       dentityConfig, Humana.ApplicationBlocks.Security"
       redirectPage="securedlogons.humana.com/common/dialogs/errordialog.asp?errms
       gid=">

       <identityConfiguration>
              <portOfOriginConfig portOfOrigin="22">
                     <businessFunctionGenKey>368</businessFunctionGenKey>
                     <businessFunctionGenKey>1107</businessFunctionGenKey>
                     <businessFunctionGenKey>1449</businessFunctionGenKey>
                     <businessFunctionGenKey>1450</businessFunctionGenKey>
                    ……..
              </portOfOriginConfig>
              <portOfOriginConfig portOfOrigin="19">
                     <businessFunctionGenKey>2792</businessFunctionGenKey>
                     <businessFunctionGenKey>2806</businessFunctionGenKey>
                     <businessFunctionGenKey>2791</businessFunctionGenKey>
                    ……..
              </portOfOriginConfig>
              <!—More PortOfOriginConfig sections as needed →

              <!—OR if you want to configure using the system application ID, do
              the following →
              <portOfOriginConfig portOfOrigin="30">

                     <systemApplicationId>33</systemApplicationId>

              </portOfOriginConfig>

       </identityConfiguration>
</humana.security>
```

The humana.security attributes determine the Identity type that the SecurityPrinciple will use the configuration class to use, and the SL redirect page. The redirect page is environment dependent, and the correct redirect page will be built from the securedLogonsConfiguration.SecurityEnvironment and the redirect page entry. Please note that the values in the IdentityConfiguration are specific to your application. Please consult with Web Security and Personalization team to obtain these values.

An application can configure security in one of two ways:
1. By listing out each business function used by the application.
2. By specifying a system application id.

System application ids are a little known SecuredLogons concept that provides a convenient means of business function grouping. Let's say, that your HSS application has five business

functions associated with it, numbered 1001, 1002, 1003,1004 and 1005. Further, let's assume that these business functions are grouped under system application id 100, and then the following two identityConfiguration sections would secure the app in the same way.

```
<identityConfiguration

       <portOfOriginConfig portOfOrigin="22">

              <systemApplicationId>100</systemApplicationId>

       </portOfOriginConfig>

</identityConfiguration>

       <!--  OR  →

<identityConfiguration

       <portOfOriginConfig portOfOrigin="22">

              <businessFunctionGenKey>1001</businessFunctionGenKey>

              <businessFunctionGenKey>1002</businessFunctionGenKey>

              <businessFunctionGenKey>1003</businessFunctionGenKey>

              <businessFunctionGenKey>1004</businessFunctionGenKey>

              <businessFunctionGenKey>1005</businessFunctionGenKey>

       </portOfOriginConfig>

   </identityConfiguration>
```

Multiple ports of origin may be specified if your application serves multiple audiences. Just repeat the <portOfOriginConfig> section for one needed,
changing the portOfOrigin attribute and contained business functions / system application ids appropriately.

Users will be allowed access to your application if they have been granted any of the business functions listed, for the port of origin found to be associated with the current session.

The Security Block has a dependency on the humana.environment configuration section. This configuration section is where you set the actual environment you are in, such as DEV, TEST, INT, QA or PROD. To add this configuration section, add the following in to the configSections:

```
<section name="humana.environment"
       type="Humana.ApplicationBlocks.Common.Config.AppEnvironmentConfigHandler,
       Humana.ApplicationBlocks.Common" />
```

Then add the configuration section:

```
<humana.environment type="DEV, TEST, INT, QA or PROD" />
```

Once the Security Block is configured and running in an application, the main interface with the application is via the SecurityPrincipal and the SecurityIdentity. The SecurityModule makes the SecurityPrinciple available on the current thread, so it is available from anywhere within an application (on the same logical tier).

To obtain the SecurityPrincipal, do the following:

```
SecurityPrincipal securityPrincipal
      = (SecurityPrincipal) System.Threading.Thread.CurrentPrincipal;
```

Once the SecurityPrincipal has been retrieved from the current thread, then the SecurityIdentity can be obtained from the SecurityPrincipal as it is a property of the SecurityPrincipal.

```
SecurityIdentity securityIdentity
      = (SecurityIdentity) SecurityPrincipal.Identity;
```

As stated above, the SecurityIdentity is the Identity of the SecurityPrincipal, so it contains a number of properties of the user. Here are a few:

```
securityIdentity.AkaName

securityIdentity.UserID

securityIdentity.FirstName

securityIdentity.LastName

securityIdentity.MiddleInitial

securityIdentity.Address1

securityIdentity.Address2

securityIdentity.City

securityIdentity.State

securityIdentity.ZipCode

securityIdentity.ZipCodePlus4

securityIdentity.TelephoneNumber

securityIdentity.TelephoneExtension

securityIdentity.FaxNumber

securityIdentity.EmailAddress

securityIdentity.UserMenuXML
```

Please see the online documentation for a description of these and other properties of the SecurityIdentity.

Obtaining Information about a user's Organizations in an Application

Every SecuredLogons user is a member of at least one organization. In the case of members and associates, there is never more than one. Providers are an example of multi-organization users. A person might do work on behalf of multiple doctors' offices. In SL, this person would have a single account and a relationship with each of the entities for whom they did work.

Each organization is represented as an object of type SLOrganization and contains demographic information about the entity as well as a list of **Access Identifiers** in the form of a collection named SLAccessIdentifiers. SL organizations can have any number of **access identifiers** associated with them. Access Identifiers store information such as federal tax ids for providers, group numbers for employers, the member gen key for members and the AIN (associate id number) for associates.

The following is a code example, showing how to iterate across a user's organizations and access identifiers.

```
// First, the requisite using statements

using Humana.ApplicationBlocks.Security;

using Humana.ApplicationBlocks.Security.DataTransferObjects;

……..

SecurityPrincipal prin = System.Threading.Thread.CurrentPrincipal as
SecurityPrincipal;


if ( prin == null )

       Response.Write ("Security principal object could not be cast to
       SecurityPrincipal.");

else

{

       SecurityIdentity iden = prin.Identity as SecurityIdentity;

       Response.Write("<b>Organizations</b>");

       foreach ( SLOrganization org in iden.Organizations )

       {

       Response.Write("<br>   Org id :" + org.OrganizationId);

       Response.Write("<br>   Org Name :" +

                        org.OrganizationName);

       Response.Write("<p>");

       Response.Write("<b>   Access Ids</b>");
```

```
        foreach ( SLAccessIdentifier accessId in org.AccessIds )
        {
        Response.Write("<br>Access Id Type : " + accessId.TypeCode);
        Response.Write("<br>Access Id :" + accessId.AccessId);
        }
        Response.Write("</p>");
}
```

Best Practices

It is recommended that as a best practice, the SecurityIdentity should be made a property of your controller so every page in a navigation graph has easy access to the SecurityIdentity. Here's a good example:

```
/// <summary>
/// Constructor
/// </summary>
/// <param name="navigator"></param>
public SomeController(Navigator navigator) : base(navigator)
{
    _currentUser
        = (SecurityIdentity)System.Threading.Thread.CurrentPrincipal.Identity;
    InitializeValuesFromState();
}
public SecurityIdentity CurrentUser
{
     get { return _currentUser; }
}
```

To get the controller in a page:

```
SecurityIdentity securityIdentity = someController.CurrentUser;
```

Using Authorization

Provide links to authorized tasks only. Please see the example below:

```
using Humana.ApplicationBlocks.Security;

const string ADD_SUBSCRIBER_TASK_NAME = "Add Subscriber";

SecurityPrincipal securityPrincipal = (SecurityPrincipal)
System.Threading.Thread.CurrentPrincipal;

if (!securityPrincipal.IsAuthorizedTask(ADD_SUBSCRIBER_TASK_NAME))

{

      lnkAddSubscriber.Visible = false;

}

else

{

      lnkAddSubscriber.Visible = true;

}
```

Make sure the authorization is checked again when the task itself is launched:

```
SecurityPrincipal securityPrincipal = (SecurityPrincipal)
System.Threading.Thread.CurrentPrincipal;


if (securityPrincipal.IsAuthorizedTask(ADD_SUBSCRIBER_TASK_NAME))

{

      UIPManager.StartNavigationTask(ADD_SUBSCRIBER_TASK_NAME);

}
```

If an operation under a task is secured, make sure the authenticated user is authorized to perform that operation:

```
SecurityPrincipal securityPrincipal = (SecurityPrincipal)
System.Threading.Thread.CurrentPrincipal;

if (securityPrincipal.IsAuthorizedTask("<Operation Name>"))

{

      … do the operation

}
```

Testing with the Security Application Block

Once your application is in a state to test, you will have to do the following:

1. Start your application in debug mode. When the start page of your application opens, you will be redirected to the SL error page (if everything is working as expected) since you do not have a valid SL session.
2. Go to test (test-hss for intranet, or one of the test internet portals) and login. This establishes a valid SL session.
3. After logging in, enter the url for your application running on your desktop. For example, if your application is called SomeApplication, enter "http://localhost.humana.com/SomeApplication/startpage.aspx" then press the enter key. You should be able to access your application if your SL credentials are correct. Please note it is very important to place the ".humana.com" on "localhost" on the url or the browser will not pick up the SL session id from the cookie.

Once you have the application running in debug mode, you should be able to exercise the application to debug as usual.

Dependencies

The Security Block has dependencies on:

- Humana.ApplicationBlocks.Common
- Humana.ApplicationBlocks.WebLibrary.

The latter is for logging. Logging has its own configuration section; please see the corresponding documentation on how to configure it.