Scanduc|23

Active Directory: Single Sign On

Jakob Kruse

Demo!

How?

- Security Assertion Markup Language (SAML) 2.0
- Comes in two parts
  - The “Service Provider” (SP), which is a web need of authentication services

    Shibboleth SP
  - The “Identity Provider” (IdP), which is a central authentication service

    Active Directory FS
- SAML establishes trust between SP and IdP, and passes authentication information between them

Prerequisites

- Your application must be deployed on a secure site (HTTPS)
- You should also have a trusted signing/encryption certificate for your application (PEM format)
  - Your Active Directory manager can probably issue one to you

Implementation

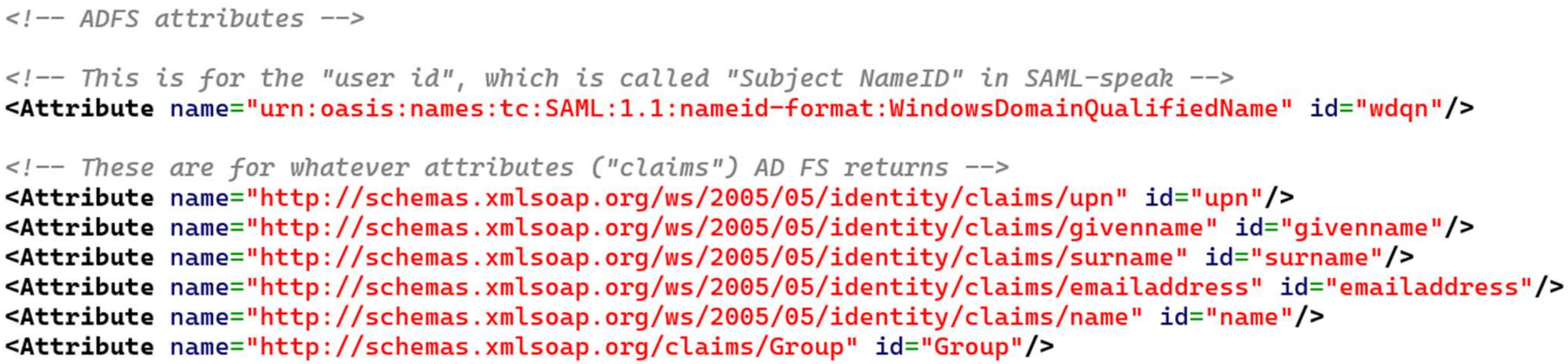
- Three steps:
  - Setup Service Provider using Shibboleth
  - Setup Identity Provider using Active Directory Federation Services (AD FS)
  - Replace authentication in DataFlex WebApp

Implementation – Service Provider

- Setup Service Provider using Shibboleth
  - Download Shibboleth SP and install on application server (master server if using SPLF)
    - Default installation + tick “Configure IIS support”
  - Place copy of FederationMetadata.xml from AD FS in C:\opt\shibboleth-sp\etc\shibboleth
  - Place copy of signing/encryption certificate in same folder
  - Configure Shibboleth SP for your application
  - Download SP metadata (from /Shibboleth.sso/Metadata)

Implementation – Service Provider

- Changes to attribute-map.xml:
  - Add attributes as needed, these are a good start ("name" is what the attribute is called in the SAML response, "id" is the name of the server variable it is put into):



```
<!-- ADFS attributes -->

<!-- This is for the "user id", which is called "Subject NameID" in SAML-speak -->
<Attribute name="urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName" id="wdqn"/>

<!-- These are for whatever attributes ("claims") AD FS returns -->
<Attribute name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn" id="upn"/>
<Attribute name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" id="givenname"/>
<Attribute name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" id="surname"/>
<Attribute name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" id="emailaddress"/>
<Attribute name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name" id="name"/>
<Attribute name="http://schemas.xmlsoap.org/claims/Group" id="Group"/>
```

Implementation – Service Provider

- Changes to shibboleth2.xml:
  - **InProcess / ISAPI / Site**
    - change id and name to match your IIS setup
  - **RequestMapper / RequestMap / Host**
    - change name and adjust Path sub-elements as needed
  - **ApplicationDefaults**
    - change entityID and add ”wdqn” to REMOTE_USER
  - **ApplicationDefaults / Sessions / SSO**
    - change entityID to match AD FS metadata
  - **ApplicationDefaults / MetadataProvider**
    - uncomment ”locally maintained” and point path to AD FS metadata
  - **ApplicationDefaults / CredentialResolver**
    - replace with reference to signing/encryption certificate

Implementation – Service Provider

- Restart your ”Shibboleth Daemon” service
- Download SP metadata from <your_application_server>/Shibboleth.sso/Metadata

Implementation – Identity Provider

- Setup Identity Provider using Active Directory Federation Services (AD FS)
  - Add a “Claims aware Relying Party Trust” under AD FS Management
  - Import data about the relying party from SP metadata
  - Configure the claims (attributes) to return on login

- To restrict access to a certain user group, go to “Edit Access Control Policy” and change the rule

Authentication Flow

- User without session requests your application
- Shibboleth redirects user to AD FS
- AD FS authenticates user (from domain login)
- AD FS redirects user back to Shibboleth
- Shibboleth stores Assertion (proof of authentication) in session
- Shibboleth puts user attributes in server variables
- Shibboleth redirects user to your application

Effect

- Only authenticated users will ever arrive at your application
- You can read information about the authenticated user from server variables and use it to configure the DataFlex session

Implementation – DataFlex WebApp

- Replace authentication in DataFlex WebApp
  - Make application do (DataFlex) auto-login on load
  - Change session manager to auto-create unknown users
  - Remove the sign out menu item and login dialog
  - Might want to increase the size of LoginName column

- This changes the meaning of WebAppUser from “users that can access the application” to “users that logged into the application at some point”!

Implementation – DataFlex WebApp

Add auto-login (WebApp.src)

```
Object oWebApp is a cWebApp
    ⋮
    Procedure OnLoad
        Boolean bOk
        String sLoginName
        Get ServerVariable of ghoWebServiceDispatcher "REMOTE_USER" to sLoginName
        Get UserLogin of ghoWebSessionManager sLoginName "" to bOk
    End_Procedure
    ⋮
    Object oHeaderPanel is a cWebPanel
        Object oMenuPanel is a cWebPanel
            Object oMenuButton is a cWebMenuButton
                ⋮
                Object oSignOut_itm is a cWebMenuItem
                    Set psCaption to "Sign Out"
                    ⋮
                End_Object
            End_Object
        End_Object
        ⋮
    End_Object
    ⋮
    Use Login.wo
    ⋮
End_Object

Send StartWebApp of oWebApp
```

Implementation – DataFlex WebApp

Change session manager (SessionManager.wo)

```
Use cWebSessionManagerStandard.pkg

Object oSessionManager is a cWebSessionManagerStandard

    Function ComparePasswords String sUserPassword String sEnteredPassword Returns Boolean
        Function_Return True // Never mind the password
    End_Function

    Function CreateUserIfNotExist String sLoginName Returns Boolean
        ⋮
    End_Function

    Function UserLogin String sLoginName String sPassword Returns Boolean
        Boolean bOk
        Get CreateUserIfNotExist sLoginName to bOk
        If (bOk) Begin
            Forward Get UserLogin sLoginName sPassword to bOk
        End
        Function_Return bOk
    End_Function

End_Object
```

Implementation – DataFlex WebApp

Change session manager (SessionManager.wo)

```
Function CreateUserIfNotExist String sLoginName Returns Boolean
    Boolean bOk
    Handle hoUserDD
    String sFullName
    Clear WebAppUser // Does the user exist?
    Move sLoginName to WebAppUser.LoginName
    Find EQ WebAppUser.LoginName
    Move (Found) to bOk
    If (not(bOk)) Begin // Create user
        Get ServerVariable of ghoWebServiceDispatcher "name" to sFullName // depends on AD FS configuration
        Get phoUserDD to hoUserDD
        Send Clear of hoUserDD
        Set Field_Changed_Value of hoUserDD Field WebAppUser.LoginName to sLoginName
        Set Field_Changed_Value of hoUserDD Field WebAppUser.FullName to sFullName
        Send Request_Save of hoUserDD
        Move (not(Should_Save(hoUserDD))) to bOk
    End
    Function_Return bOk
End_Function
```

Where?

- Internal web-based business applications

Where application users are known in advance and are already logged on to a domain / Active Directory.

Why?

- Improves user experience
  - Single Sign On, skipping application login altogether
  - Don’t have to remember another password
- Increases security
  - No credentials stored in application
  - Leaving authentication to the experts
- Compliance
  - Centralized user/rights management is often a requirement

Remember!

- User ID and attributes depend on AD FS configuration
  - These are called “Claims” in AD FS
- Shibboleth SP needs configuration to read AD FS claims
  - This is what we configured in “attribute-map.xml”

Also…

- If you have more than one application:
  - Group by access rules
  - Put each group on separate virtual hosts
  - One installation of Shibboleth SP can be configured for multiple virtual hosts / service providers
- Helpful tool while testing:
  - https://www.samltool.com/

Scanduc|23

Thank you!

Are there any questions?