Symplified I: Windows User Identity

Matthew McNew and Lex Hubbard

Table of Contents

Abstract

Single Sign On (SSO) allows users to access multiple services with only one login instead of having to repetitively enter their username and password at every prompt. SSO curtails phishing attempts, prevents password fatigue, and decreases calls to Information Technology (IT) Departments. This project involves enabling SSO through an Identity Router (IdR) that exists in a separate domain from Windows Active Directory.

The company, Symplified, allows SSO to cloud applications through their IdR. The goal of this project is to create software that Symplified's clients can deploy. This software (referred to from here as SimpleIWA) authenticates Windows users from the client's Active Directory (AD) domain to the IdR that exists in a separate domain. SimpleIWA pulls user information from an Active Directory on a separate domain from Symplified's on site IdR. SimpleIWA installs an ASP.Net page that exists on a Windows Internet Information Services (IIS) server and processes requests using Integrated Windows Authentication (IWA). At the user's request, SimpleIWA pulls Windows user information from AD and a signed Security Access Markup Language (SAML) document is generated and sent to the IdR. This process is seamless without any user interaction and it is invisible to the user.

SimpleIWA has to be deployed to client IIS servers. Therefore SimpleIWA is packaged for distribution with a configurable Visual Studio Installer. SimpleIWA has been tested and documented with different configurations.

Introduction to the Project

Symplified provides its clients with an IdR that allows secure access to cloud applications through SSO. Symplifed's IdR, known as Single Point, authenticates users on the client's network through Active Directory. The current system requires that the IdR be in the same domain as the client's Active Directory and many of Symplified's clients would prefer to create a separate domain for the IdR.

Project Description

The solution was named SimpleIWA. With SimpleIWA, Symplifed's IdR redirects a user's browser to the SimpleIWA ASP.Net page on the client's IIS server. SimpleIWA utilizes IWA to identify the user's information. SimpleIWA pulls user attributes from Active Directory including the user's Username and any other attributes the client prefers to include in the response. Then SimpleIWA generates a SAML document with the user information and any additional attributes. Before submitting the document a configuration file is read that includes an Assertion Customer Service URL (ACS URL), a Relay State, and the location of the private key. The SAML document is signed with the private key and it is submitted through the user's browser to the ACS URL on the IdR.

SimpleIWA is deployed using a Visual Studio Installer. The installer checks for IIS, .Net, and IWA. It offers links to tutorials and download pages if these requirements are not present on the current system. Once these requirements are confirmed, SimpleIWA Installer enables the correct .Net version for the webpage to run under. The installer also requests information from the user. The first tab has fields for the requirements: an ACS URL, an Issuer, an Audience, and a Relay State. The second tab contains a check box list for LDAP fields to be pulled from AD for authentication. The last tab allows the user to provide the private key location.

Requirements Specification

Functional Requirements

SimpleIWA must process user identity from IWA and create a SAML assertion that will be sent to an Identity Router. In addition SimpleIWA must be packaged allowing it to be easily distributed. Specific requirements include:

- Maintain the current system's SSO functionality
- Accept user browser requests
- Identify Active Directory users with IWA
- Require the user to log into Active Directory if the user identity cannot be provided by IWA
- Create a SAML assertion that verifies the user and provides other, to be determined attributes of that user

- Send the SAML assertion to the IdR over HTTP Post
- Create an installer to install the product on different servers
- Allow the application to be configured in the installer

Non-Functional Requirements

- Allow SimpleIWA to be deployed on any domain within the network
- Write SimpleIWA in C# with ASP.Net
- SimpleIWA must install and run on a Windows Server running IIS

Architecture Diagram

Use Cases

SimpleIWA will allow a specific configuration: the client's Active Directory will be able to send user authentication to the IdR that Symplified provides while the IdR is in a separate domain.

Installation

1. Display installation screen
    A. Prompt for Install
        i. Verify IWA, IIS, and ASP.Net
            a. Provide links and installation information
        ii. Enable most recent version of .Net in IIS
        iii. Prompt for configuration information
        iv. Exit
    B. Prompt for Exit
        i. Exit
2. SimpleIWA is installed on Server waiting for requests

Single Sign On Authentication

1. User logs on to Active Directory with username/password combination
2. User's browser requests SimpleIWA
    A. Software authenticates the user
    B. User is redirected to the intended location

Authentication With User Not Logged into Active Directory.

1. User's browser requests SimpleIWA
    A. Software asks the user for username/password combination
    B. User logs in with Active Directory username/password combination
        c. User is redirected to the intended location
2. User enters incorrect username/password combination
    A. Software asks for username/password combination again

Design

Design Summary

SimpleIWA contains two distinct components. The first component is the Default ASP.NET webpage written in C# that generates the SAML assertion. The second component is the installer and configuration utility.

The Default ASP.NET Web Page sets up the web page and calls the other processes when the page is loaded. The modules in this web page are outlined in Figure 1. Module A reads information on the ACS URL, relay state, and the location of the private key from a configuration file. Module B identifies the user and pulls information from Active Directory. Module C takes the user information from the first process and generates a signed and base 64 encoded SAML Assertion. The default ASP.NET webpage sends the assertion and relay state over HTTP POST to the ACS URL.

The installer installs and configures the application for IIS. The modules of the installer are described in Figure 2. A Visual Studio Installer project is used to install the application and detect dependencies. The installer utilizes a C# Custom Action to configure and setup the application. This Custom Action can also be separated into modules. Write Modules. Modules: Configure IIS, Config Form,

ASP.NET Web Page Design

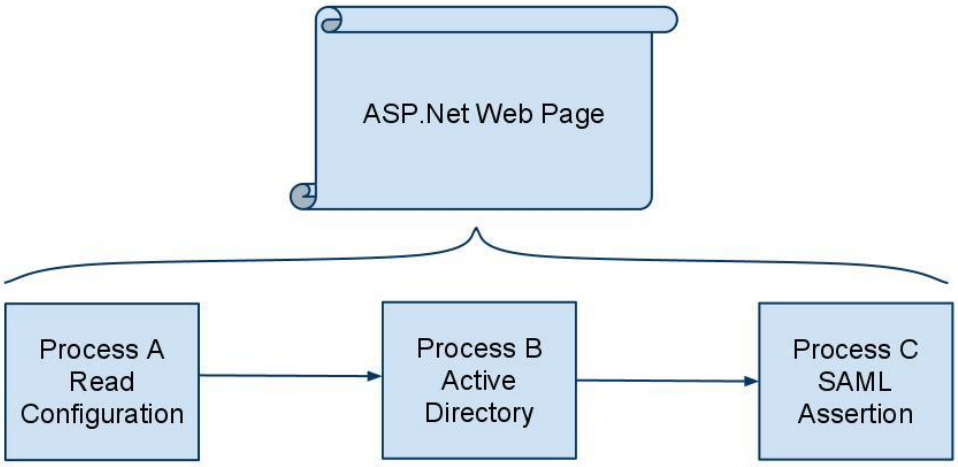

Figure 1. The Components of SimpleIWA

Default ASP.NET Web Page

The Default ASP.NET Web Page is called on the server. This web page specifies an XHTML Splash Screen for the user. The Page Load function calls the Read Configuration Module to load the SimpleIWA configuration and the Determine User Information from Active Directory Module to identify user information. Using this information, a SAML assertion is created with the Generate SAML Assertion Module. Then an XHTML form that has two attributes is created. One attribute is the RelayState. The other attribute is the Base 64 encoded SAML document. Javascript is used to automatically submit the form to the ACS URL. The Relay State and the ACS URL are obtained from the Configuration File.

Module A - Read Configuration

There is an XML configuration file in the root of the ASP.Net application. This module reads the configuration file and obtains important configuration details for other components. Existing .Net XML serialization libraries are used to read and parse the file. The configuration file contains five unique fields. One field describes the ACS URL which is where SimpleIWA submits the SAML assertion. Another field describes the relay state of application. This field is also used when the SAML assertion is submitted and redirects the browser. Another field

describes the location of the private key. This configuration detail is used when the SAML assertion is digitally signed in the Generate SAML Assertion module. A field describes the Issuer which informs the IdR of the SAML Assertion origin. And another field describes the NameID which is the format of the username.

Module B - Determine User information from Active Directory

SimpleIWA pulls user information from Active Directory. The information pulled includes the user's login name and any information Symplified would like to use to authenticate that user to the IdR. The required attributes to pull from Active Directory are specified in the configuration file. SimpleIWA pulls these fields from LDAP (Lightweight Directory Access Protocol) using a modular function on each requested field. In addition to the user's login, SimpleIWA is able to authenticate the user by pulling their information from AD such as the full name, email address, memberships, etc.

Module C - Generate SAML Assertion

The SAML assertion is generated with the user information that is provided by Active Directory in Module B. To generate the SAML document, existing Code Project libraries were modified to reduce unnecessary work. Given attributes, a SAML XML document of type XmlDocument in System.Xml is generated. The private key is read and opened. Existing libraries in System.Xml.Cryptography are used to generate a digital signature with the private key. Then the XML document and the Signature are Base 64 bit encoded. The product of this operation is used by the ASP.NET webpage to submit the SAML Assertion.

Installer and Configuration Utility Design

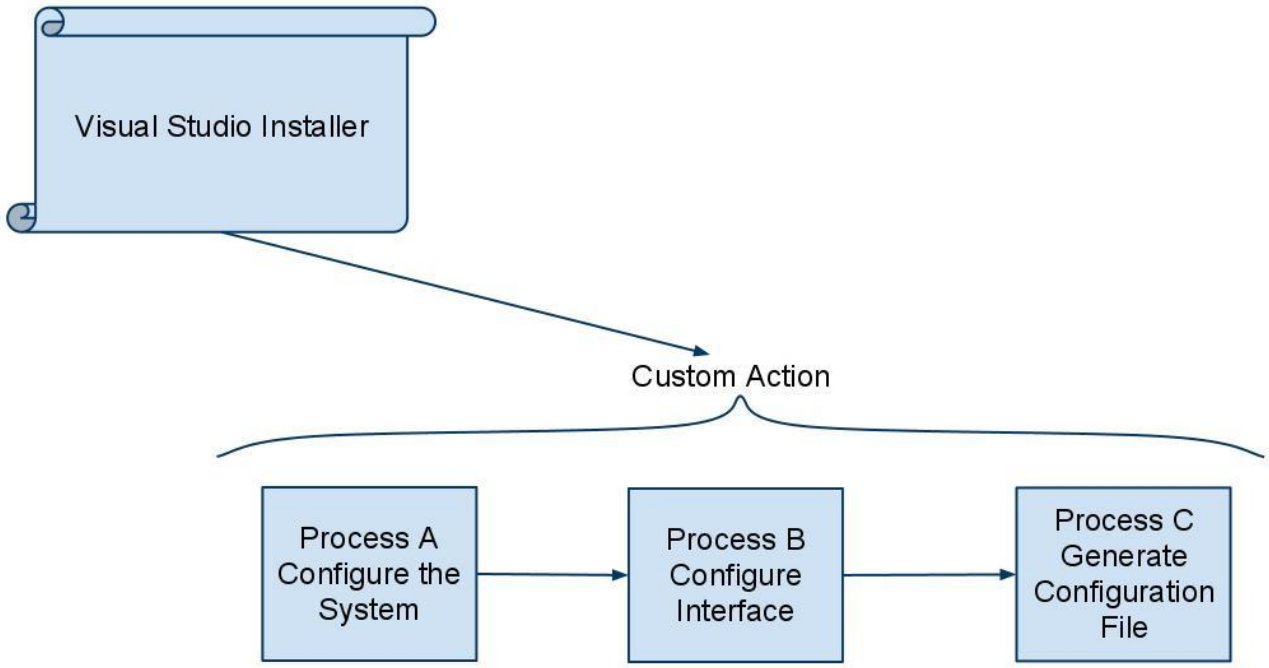

Figure 2 The Components of the SimpleIWA Installer

Virtual Studio Installer

The Virtual Studio Installer is configured using the Virtual Studio Interface. The installer checks for dependencies by querying the Windows Registry for the required registry entries. If a dependency is missing, instructions are provided on how to install the necessary dependency. A branded interface in the installer allows the user to configure where SimpleIWA will be installed and which AppPool to use on IIS. After these preliminary configuration details have been set up SimpleIWA is installed at the specified path. At the end of the install process a Custom Action is executed. This Custom Action configures the system and provides a User Interface to configure the SimpleIWA configuration file. This Custom Action is described in greater detail below in three different processes.

Process A - Configure the System for SimpleIWA

At first, the Custom Action configures the System for SimpleIWA. This is done by identifying the version of IIS on which the installation is being performed. On IIS 6 dynamic content is enabled. On IIS 7 ISAPI restrictions are removed. On all platforms the correct version of .NET is set on the AppPool running SimpleIWA. In addition, Windows Authentication is enabled in IIS and the Network Service user account is given write permissions to the Windows Temp Directory.

Process B - Configuration Interface

A tabbed Windows Form is displayed to provide an interface to configure the SimpleIWA configuration file. The interface allows users to specify the ACS URL, the Relay State, the Audience, the Issuer, and the SAML Subject. A different tab allows the users to specify which Active Directory attributes to include in the SAML assertion. A third tab allows users to choose a Certificate containing a private key with a file section dialog. The file selection dialog runs on a separate thread from the installer. When the form is submitted, Process C takes place and the Configuration File is Generated

Process C - Generate Configuration File

The configuration file is generated using the C# XML serialization library. Each field of the form is read and added to the XML document as an element of the appropriate type. Each Active Directory attribute from the checkbox list is used as a key to find the appropriate LDAP attribute value in the LDAP dictionary inside of the Windows Form. The location of the private key is requested and both the key and the configuration file are added to SimpleIWA's installation location.

UML Diagrams

Two simplified UML diagrams are shown below. The diagram, in Figure 3 illustrates the different modules in the SimpleIWA ASP.NET webpage. The user's web browser is directed to the Default WebPage. When this happens the Page_Load function is executed. This function initializes and utilizes the other classes as described in detail above. The ConfigFile class is used to obtain configuration attributes. The UserInfo class is utilized to obtain IWA user identity information. The SamlBuilder class is used to generate a SAML assertion.

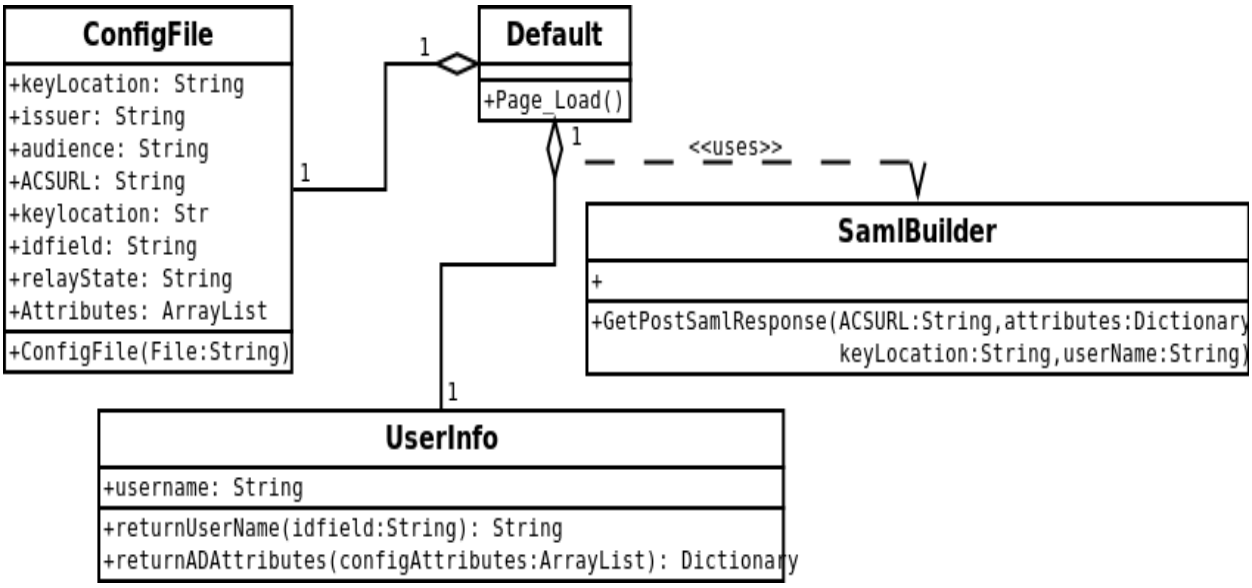

Figure 3 UML Diagram for the SimpleIWA ASP.NET webpage

Figure 4 illustrates the basic design of the SimpleIWA Installer Custom Action. The Custom Action class is used by the Visual Studio Installer. At the end of the install process the Installer function is executed. This function configures the System for the SimpleIWA application. The function also initializes a ConfigForm that allows the user to configure SimpleIWA. The ConfigForm class uses the ConfigWriter to write the SimpleIWA configuration file.

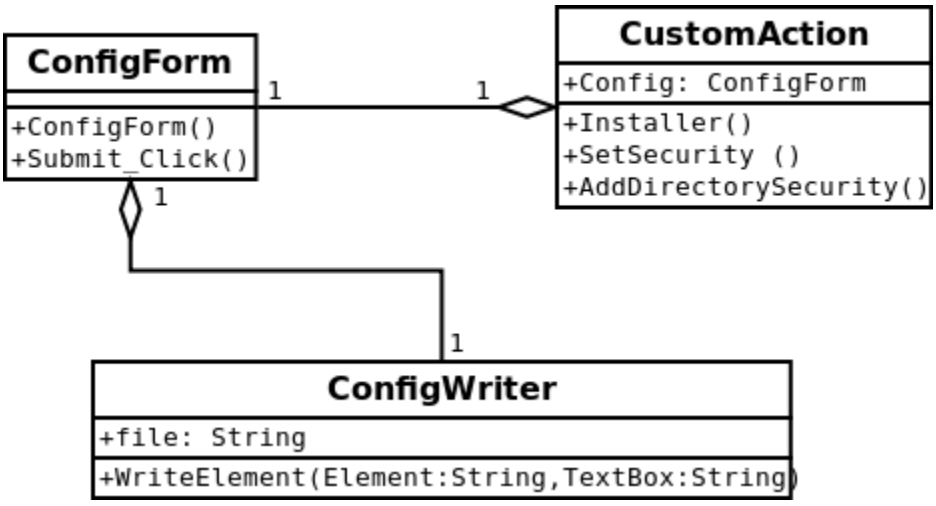

Figure 4 UML Diagram for the SimpleIWA Installer Custom Action

Implementation Details and Results

SimpleIWA was written in Visual Studio 2010 using C# and ASP.Net. Windows IIS uses virtual directories that run ASP.Net web pages and there was quite a lot of functionality for SimpleIWA's requirements already existing in Visual Studio C#.

The configuration file follows an XML schema. This was a natural choice for the configuration file because it follows the pattern of other IIS and ASP.NET configuration files. In addition, SimpleIWA is required to produce a SAML Assertion, a protocol that is based on XML.

SimpleIWA utilizes a modified SAML Library written in C# and provided by The Code Project Open License. A library was used to generate SAML to avoid rewriting publicly available code. This library was chosen because it was well documented. In addition, the simplicity of the library allowed it to be modified and integrated into SimpleIWA.

The private key file chooser dialog in the configuration utility runs on a separate thread from the installer thread. This is because the Windows OpenFileDialog class is required to run on a thread with a single threaded apartment state. The installer thread does not satisfy this requirement. A separate thread is created to use the Windows class and provide familiar functionality to the user.

Windows Installer XML (WIX) was considered as the basis for SimpleIWA's installer, but after research, it was determined that the learning curve for WIX would be too steep. Ultimately, A Visual Studio Installer fit the scope of the project. A Visual Studio Installer is not as powerful as using a WIX installer. However, it offers a complete ASP.NET installer that can be configured with Custom Actions to add additional functionality. This additional functionality was essential to allow SimpleIWA to be configured in the install process.

Integration Testing

SimpleIWA was tested with six different Active Directory servers. The servers were running on different versions of Windows and had different releases of IIS. In addition SimpleIWA was tested with Symplified's Single Point Studio Software. SimpleIWA works smoothly in all the different testing environments. Many of the different testing environments have different dependencies. The SimpleIWA installer identifies when these dependencies are missing and provides instructions to install them.

The SimpleIWA application requires .NET version 4. IIS is also required. However, only IIS versions 6 and 7 have been tested. During the installer the AppPool and the virtual directory are specified. The installer will enable dynamic content on the specified AppPool. When using IIS 7 a few different role services are required. The Windows Authentication role service and

compatibility components role service must be installed. When these dependencies are not installed the installer provides instructions on how to install them.

Servers

SimpleIWA Installer and SimpleIWA functionality were tested on the following servers through a Remote Desktop Connection.

Windows 2003 DC with IIS 6.0 (testlified.com)
VNC beavis.bdr.symplified.net:5922

Windows 2003 DC with IIS 6.0 (symp.testlified.com)
VNC beavis.bdr.symplified.net:5929

Windows 2008 DC with IIS 7.0 (symplified.sqa)
VNC beavis.bdr.symplified.net:5915

Windows 2008 DC with IIS 7.0 (qa1.symplified.sqa)
VNC beavis.bdr.symplified.net:5917

Windows 2008 DC with IIS 7.0 (qa1.symplified.sqa)
VNC beavis.bdr.symplified.net:5954

Windows 2008 DC with IIS 7.0 (topherific.com)
Remote Desktop 10.200.20.80

Clients

SimpleIWA functionality was tested with the following clients through a VNC connection.

Windows Vista Business
VNC beavis.bdr.symplified.net:5950

Windows 7 Professional
VNC beavis.bdr.symplified.net:5964

Windows XP Proffessional
VNC beavis.bdr.symplified.net:5938

Windows 7 Professional
Remote Desktop 10.200.20.81

Scope and Project Progression

SSO functionality was required. Using IWA to verify Active Directory users was required. SimpleIWA had to create a SAML assertion that would allow the IdR to verify a user's access to cloud applications. The IdR had to be able to do this from a separate domain than the client's Active Directory domain. Clients had to be able to deploy SimpleIWA on any domain in their network.

Symplified required an installation UI for SimpleIWA. Ultimately, time allowed for this to fall into the scope of this project.

SimpleIWA met all of the goals that Symplified set out for it to accomplish. One of the main aspects of SimpleIWA was creating an application that would make it easier for Symplified's clients to obtain the functionality they desired through Symplified's service. With that in mind, additional functionality was added to the product to make it more user friendly, especially with the installation. SimpleIWA deals with IIS in a user friendly way by providing helpful links and installing necessary components itself. It also reconfigures IIS on its own to make the experience as automated as possible for the client administrators. In addition, SimpleIWA is very easy to reconfigure. The configuration form can be opened and resubmitted at any time after SimpleIWA is installed.

Conclusion

SimpleIWA is a product with massive potential to simplify SSO tasks for users operating on Windows Networks. The software has been demonstrated to Symplified's Development and Marketing Team. These employees are very excited to bring the product to Symplified's clients.

Some desired functionality was outside of the scope of the project. It would be beneficial for SimpleIWA to log access requests to enable more effective system maintenance. In addition a more portable installer could be created using WIX. This installer could have greater functionality and be easily expanded to different versions of Windows and IIS.

Even without the functionality that was outside the project scope, SimpleIWA development used many different and useful technologies. We realized that Visual Studio can be used to collaborate multidimensional products. We learned that SAML has a wide range of useful functionalities that can be utilized in many Internet applications to provide an easier method of authentication. Finally, we discovered that free fall design techniques are difficult to work this and pursuing an agile philosophy can speed up and enhance development.

Glossary

Active Directory (AD) - Created by Microsoft, provides many services to a network, a central service for administrators to reside over a network and provide security
Active Server Pages (ASP) - A server side script engine for dynamically generated web applications
ASP.Net - Web application framework for building web applications
Assertion Consumer Service URL (ACS URL) - The location where the SAML assertion is sent, the Service Provider's authentication location
Extensible Markup Language (XML) - A protocol for creating documents that are easily read by machines
HTTP (Hypertext Transfer Protocol) - A networking protocol and the basis for the Worldwide Web
HTTP POST - An HTTP request method that allows for an arbitrary length of data
Identity Provider (IDP) - A service that registers URL identities and provides authentication
Identity Router (IdR) - A device that allows internal network identities to be associated with external network identities
Internet Information Services (IIS) - Web server application by Microsoft. Administration Pack adds ASP.Net authorization
Integrated Windows Authentication (IWA) - Allows automatic authentication between Active Directory aware applications
LDAP (Lightweight Directory Access Protocol) - A network directory protocol for accessing and editing information over TCP/IP
Security Assertion Markup Language (SAML) - An XML based publicly available standard for authenticating across domains between an Identity Provider which provides assertions and a Service Provider which consumes them
Service Provider (SP) - An entity that provides a service, a business that provides a server (either hardware or virtual), an entity that hosts a web application
Single Sign On (SSO) - The ability for a system to verify a user to multiple resources with only one prompt for a username/password combination
Software as a Service (SaaS) - A software delivery model where software is hosted on the internet (cloud) and accessed by users, typically through a web browser
Uniform Resource Locator (URL) - A Uniform Resource Identifier with a location and instructions on how to get to that location, http://www.mines.edu is an example of a URL
Windows Installer XML (WIX) - Free software that builds Windows Installers (.msi) from XML documents

References

Speight, David W. “Performing a SAML Post with C#.” *The Code Project.* March 8, 2010. May 19, 2011. <http://www.codeproject.com/KB/aspnet/DotNetSamlPost.aspx>