Journal of Mobile, Embedded and Distributed Systems, vol. III, no. 1, 2011
ISSN 2067 – 4074

Web Single Sign-On Implementation Using the SimpleSAMLphp Application

Ionut Andronache, Claudiu Nisipasiu
Military Technical Academy
George Coşbuc Blvd., no. 81-83, Sector 5,
Bucharest, zip code 050141
ROMANIA
ionut.andro@gmail.com, claudiu.nisipasiu@gmail.com

Abstract: Web Single Sign-On is a feature offered by web applications that have a trust relationship, not necessarily within the same company. The goal of Web SSO is to provide authentication information for all the web application in the trust relationship, without requiring the user to login in each web application. SAML 2.0 is the standard that defines the framework in order to achieve Web SSO and identity federation in a web context. In order to make a Web SSO implementation, we used the open-source SimpleSAMLphp library, which implements the standards of SAML 2.0 and provides functionality for the two scenarios: SAML – Service Provider and SAML – Identity Provider.

Key-Words: Web Single Sign-On, Identity Federation, Identity Provider, SAML 2.0, SimpleSAMLphp

1. Introduction

With the involvement of IT systems in public domains like banking, commerce, school, administration and so on, we assisted to the birth and fast growth of their branches from the virtual environment, branches like: e-banking, e-learning, e-commerce, etc. These new branches offer us a new safe alternative, less time-consuming for activities that, not long ago, we didn't even imagine we could do without our physical presence (paying bills, learning, shopping, voting, etc.). In order to take advantage of these benefits, the user nowadays must authenticate in a lot of systems, to prove his identity. Users have to manage a lot of accounts, to login to multiple systems, necessitating an equivalent number of sign-on dialogues to enter credentials. System administrators are faced with managing user accounts within each of the multiple systems to be accessed in a coordinated manner in order to maintain the integrity of security policy enforcement. Problems in this scenario on the user side were: users neglecting their passwords, forgetting their accounts, writing them down on papers, the usage of weak passwords to remember them or the same password for more accounts, the process is time-consuming, etc. On the other hand, companies had to deal with a lot of frauds, high costs for strong authentication systems, help-desks for password recovery and so on. To solve these problems and to give other benefits, both to the users and companies, rose up two new concepts: Single Sign-On and a new approach of identity management, known now as "federated identity management".

This is a post conference paper. Parts of this paper have been published in the Proceedings of the 3rd International Conference on Security for Information Technology and Communications, SECITC 2010 Conference (printed version).

1.1 Single Sign-On (SSO)

The Open Group defines Single Sign-On *as a mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where that user has access permission, without the need to enter multiple passwords*. [1]
The most important security advantage that a SSO implementation offers is a

common secure infrastructure, which can be carefully managed and protected [3]. Think how important is to verify easily user security information and update when necessary rather than tracking down all operational systems. This is particularly valuable when users move to new roles with different access levels.
SSO was to be achieved by developing all the applications and tools to use a common security infrastructure, to have the same authentication information format, avoiding the current situation of heterogeneous security infrastructures. Creating a common enterprise security infrastructure works very fine when developing new applications, but the real concern is for existing systems. Unfortunately, the task of changing all existing applications to use a common security infrastructure is very difficult. Because of these difficulties in having a single security infrastructure, many enterprises use a so called proxy-SSO solution [2]. A proxy-SSO product authenticates to existing legacy systems without user intervention. It uses a proprietary authentication mechanism and then re-authenticates transparently to multiple underlying systems via the user interfaces to those systems. This involves a scripting engine that drives the interaction with each underlying system's authentication challenge. Custom scripts must be developed for every possible user authentication interaction and also when the underlying systems change their interfaces, the scripts require changes. The proxy-SSO solution is difficult, expensive and time-consuming. Also from the security point of view, the company still has multiple security infrastructures and now has one more security product to look after (a new point of attack). The best solution, which offers a lot of benefits to the users and saves a lot of money from the companies, is the SSO implementation with a unique user registry. Using a common registry requires that all existing applications be migrated to use this new registry. The best registry to choose is one that supports Lightweight Directory Access Protocol (LDAP).

A new form of limited SSO is becoming popular: Web SSO.
Systems offering Web SSO have a central infrastructure, including a Directory, in order to manage Authentication and Access Control. A Directory is a collection of accounts managed by a single system, internal to a system like the the SAM database in a Windows NT domain, or shared by multiple systems, like a LDAP Directory.[4]
These systems also have an agent installed on web servers, so when users attempt to Access a Web SSO - enabled web server or web application, the Web SSO agent redirects the user's web browser to an Authentication Server, where the user signs in. The web browser is then redirected back to the requested web application, and the User can Access the application or web content When an already authenticated user accesses another web application, the agent on the web application retrieves the user's validated credentials, thus eliminating any need for the user to sign on again. Web SSO systems also incorporate Access Control mechanisms, where either the agent installed on each web server, or the web applications themselves (using an API), may check the user’s rights.

1.2 Federated Identity

Federated identity refers to the standards, tools and use-cases that serve to enable the portability of identity information between different autonomous security domains. The goal is to offer the user from one domain the possibility to securely access data from another domain, without the need for completely redundant user administration. The use of identity federation standards eliminates the need for new account registration through automatic "federated provisioning" or the need to redundantly login through cross-domain single sign-on [6].
The notion of “identity federation” is generic, is not bound to a specific protocol, technology, implementation or enterprise. It could involve user-to-user,

ISSN 2067 – 4074

user-to-application as well as application-to-application use-case scenarios at both the browser tier as well as the web services or service-oriented architecture (SOA) tier. Identity federation can involve high-trust, high-security scenarios as well as low-trust, low security scenarios. The Identity Assurance Framework is standardizes the levels of identity assurance that may be required for a given scenario. It can involve user-centric use-cases, as well as enterprise-centric use-cases.
Identity federation can be accomplished in many ways. It can be done with the aid of formal Internet standards, such as the OASIS Security Assertion Markup Language (SAML) specification, or may involve open source technologies and/or other openly published specifications (e.g. Information Cards, OpenID, the Higgins trust framework or Novell's Bandit project) [5].
In the following sections, we will present SAML 2.0, a version of the SAML OASIS standard for exchanging authentication and authorization data between security domains and SimpleSAMLphp, an application written in native PHP, having a main focus on providing support for:

- SAML 2.0 as a Service Provider.
- SAML 2.0 as a Identity Provider.

2. SAML 2.0

The Security Assertion Markup Language (SAML) standard defines a framework for exchanging security information between online business partners. It was developed by the Security Services
Technical Committee (SSTC) of the standards organization OASIS (the Organization for the Advancement of Structured Information Standards).
SAML is different from other security systems due to its approach of expressing assertions about a subject that other applications within a network can trust. [7]
The four main purposes behind the creation of the SAML standard are:

- Browser cookie limitations
- SSO interoperability
- Web services security
- Identity federation

There are two very important concepts that play an important role in the SAML standard:

- Identity Provider (IdP) – is the system or the domain that asserts that a user has been authenticated and has associated attributes. In SAML, Identity Providers are also known as SAML authorities and Asserting Parties.
- Service Provider (SP) – is the system or administrative domain that relies on information supplied by the Identity Provider.

The Service Provider decides if it trusts or not the information provided by the IdP. Service Providers are also known as Relying Parties – due to the fact that they "rely" on information provided by an Identity Provider (Asserting Party).
SAML consists of some block components that supports a number of use cases. First of all, these components permit transfer of identity, authentication, attribute and authorization information between autonomous systems that have a trust relationship. The core SAML specification defines the structure and content of both assertions and protocol messages used to transfer this information.
SAML assertions carry statements about a principal that an asserting party claims to be true. The valid structure and contents of an assertion are defined by the SAML assertion XML schema. Assertions are usually created by an asserting party based on a request of some sort from a relying party, although under certain circumstances, the assertions can be delivered to a relying party in an unsolicited manner. SAML protocol messages are used to make the SAML-defined requests and return appropriate responses. The structure and contents of these messages are defined by the SAML-defined protocol XML schema [9].
SAML supports two kinds of message flows: IdP-initiated or SP-initiated. These flows refer to the place where the user starts the process of a web SSO exchange. The most common scenario is when the user wants to access directly a resource that is on a service provider

site and the service provider sends the user to authenticate at the Identity Provider. This flow is de SP-initiated one. The IdP initiated flow is when the user is visiting an IdP where he authenticates and after that he clicks on a link to a partner SP.

2.1 SAML – Most important use cases

The most important use case for which SAML 2.0 is applied is probably Web SSO. In this use case, a user has a login session on a web site, the user accesses resources on that site and at some point he is directed on a partner's website. If a federated identity for the user was established prior to this moment between the two sites, then the first site, the Identity Provider asserts to the Service Provider, the second site, that the user is known, authenticated and also has some specific attributes. As we mentioned above, a user's identity is said to be federated between a set of providers. This happens when there is an agreement between the providers on a set of identifiers and/or identity attributes by which the sites will refer to the user. This is the federated identity use case. SAML 2.0 enhances the federated identity capabilities and introduces the high-level identity federation use case [7]. Most identity management systems maintain local identities for users. These local identities could be the user's local login account or some other locally identifiable user information. These local identities must be linked to the federated identity that will be used to represent the user when the provider interacts with a partner. The process of associating a federated identifier with the local identity at a partner (or partners) where the federated identity will be used is often called account linking. So, when a user migrates from a site where he is authenticated to another site and the two sites have a business agreement that establishes the federation of identities and also, on the second site, the user has a local identity, linked to the federated identity, the user will be recognized on the second site, and he will be referred with his local identity.

2.2 Security and privacy in SAML

SAML defines a number of security mechanisms to detect and protect against attacks like "man-in-the-middle" or replay. First thing to do is that the relying party and asserting party to have a pre-existing trust relationship which typically relies on a Public Key Infrastructure (PKI). While use of a PKI is not mandated by SAML, it is recommended. Also it is recommended the use of HTTP over SSL 3.0 or TLS 1.0 when message integrity and message confidentiality are required. When a relying party requests an assertion from an asserting party, bi-lateral authentication is required and the use of SSL 3.0 or TLS 1.0 using mutual authentication or authentication via digital signatures is recommended. It is also mandated that a response message containing an assertion, that is delivered to the relying party via a user's web browser (e.g. using the HTTP POST binding) , to be digitally signed using XML Signature in order to ensure message integrity.
In terms of privacy, SAML 2.0 supports the establishment of pseudonyms established between an identity provider and a service provider and also one-time or transient identifiers. SAML's Authentication Context mechanisms allow a user to be authenticated at a sufficient (but not more than necessary) assurance level, appropriate to the resource they may be attempting to access at some service provider [8].

3. SimpleSAMLphp

SimpleSAMLphp is an application written in PHP that implements the standards of SAML 2.0 and offers support for the two scenarios: SAML as a Service Provider and SAML as an Identity Provider [12].
In case of using the SimpleSAMLphp as a Service Provider, it will communicate

ISSN 2067 – 4074

and delegate authentication with an SAML Identity Provider.
As SimpleSAMLphp is written in PHP, it is a very simple way of integrating web-based PHP application into a federation. SimpleSAMLphp also support non-PHP environment by using the Auth Memcookie approach: a special cookie is added in memcache that the Apache module Auth MemCookie understands. The authentication information is stored in header variables and the authorization is configured in Apache.
SimpleSAMLphp as an Identity Provider can accept connections from Shibboleth and SAML 2.0 services. The Identity Provider does not store the user credentials on its own and it uses a credential source to validate against. There are several built-in authentication modules to support this, such as LDAP, CAS service, Radius authentication, SQL authentication, OpenID and YubiKey. SimpleSAMLphp offers an Extension API in order to allow customization and integration with third party modules. The flexibility supported by SimpleSAMLphp focuses on a major subjects: authentication modules – for implementing a custom authentication module, such as PKI based; authentication processing filters – to allow processing right after authentication have taken place; themes – to customize the look of the pages; SimpleSAMLphp modules – to allow the development of new identity protocols, pages, registry systems etc.
Besides the basic functionality of a Service Provider and of an Identity Provider, SimpleSAMLphp supports other advanced features, such as: bridging between protocols, attribute control, automated testing and metadata signing.
SimpleSAMLphp includes libraries for integration with several popular Content Management Systems (CMS), like Drupal, DokuWiki and MediaWiki.

3.1 Service Provider Scenario

SimpleSAMLphp as a Service Provider connects to an Identity Provider to authenticate. This can be easily defined in the configuration files.
The Service Provider can be configured to connect to multiple Identity Providers and the user will choose from a list.
In the web application, we want to implement SimpleSAMLphp as a Service Provider. We have to include the classes of SimpleSAMLphp and by using the API, we require user authentication. Once the authentication is complete, we can access the user's attributes.
The Service Provider API provides basic functionality, such as: check if the user is authenticated, require authentication, login, logout, get user attributes, get URLs for login and logout.
Since the Service Provider connects to the Identity Provider, the web application has no knowledge of the Identity Provider. So, the Service Provider can be configured to connect to other Identity Providers without having to change anything in the web application.

3.2 Identity Provider Scenario

The SimpleSAMLphp as an Identity Provider can be configured to authenticate users against various sources – static, LDAP, SQL, Radius, OpenID, YubiKey, Facebook and Twitter.
To setup the Identity Provider a few configuration files have to be changed in order to specify the authentication module used and its additional information and the list of Service Providers [10].
When several Services Providers use the same Identity Provider to authenticate the user, the user has to login only once, since his session information is stored by the Identity Provider. The Single-Sign-On mechanism can be deactivated for a specific Service Provider by adding some extra parameters in the request to the Identity Provider, including EntityID - id of the Service Provider.
The Identity Provider also requires a certificate so that he can prove its identity to the Service Provider. For test purposes, it can be used a default

certificate that ships with the SimpleSAMLphp files.

3.3 Experiment

In our experiment, we have used SimpleSAMLphp as both Service Provider and Identity Provider, in order to demonstrate its functionality for both scenarios.
In Appendix 1 we have included some scripts used for different steps in our experiment.
To make the experiment we used two computers in a local network, one as a Service Provider and the other as an Identity Provider.
On the Service Provider machine, we have installed the required applications Apache web server, PHP with the required modules and have configured the SimpleSAMLphp.
The Service Provider is used by a sample page which requires authentication. The sample page loads the SimpleSAMLphp classes and requires authentication.
To configure the Service Provider, we have to define first the authentication source. We use "default-sp" to connect to our Identity Provider on the nework – 172.16.1.11:8000 and to setup the additional metadata related to the Identity Provider, such as the urls for login/logout and a fingerprint of the Identity Provider's certificate.
The Identity Provider uses a SQL authentication module and stores the credentials in a table. This SQL authentication method called "myauth" is defined in the configuration file of the simpleSAMLphp. The Service Provider metadata is described in a different configuration file.
In order to test the Single Sign On mechanism, we created another web application (basically a simple page) deployed on a different server. Both Service Providers from the two applications were configured to use the same Identity Provider.
So, after the user accesses one of the applications and he logs in, when he goes to the second application, he shouldn't be required to login again, since both applications use the same Identity Provider.

3.4 Results

The user accesses the first application and he is redirected to the Service Provider page, where he can choose from a list of available Identity Providers.
He chooses our Identity Provider on the server in the network and he is redirected to the Identity Provider Login Page.
After he logs in, he is redirected to the Service Provider and then to the web application that he requested.
On the web application sample page, the user information is displayed correctly (same information as in our table in mysql).
The user goes to another page on the web application and his login information was stored in a session by the Service Provider so he doesn't have to login again, his information is displayed correctly.
The user accesses web application 2, which is hosted on a virtual host on the network. As mentioned earlier, the web application 2 Service Provider uses the same Identity Provider.
The user is redirected to the Service Provider page, where he can choose from a list of the available Identity Providers.
He selects the same Identity Provider and since the user already logged in for the first web application, he is redirected back to the Service Provider and from there back to the web application 2 – the page he requested.
So, the SingleSignOn mechanism worked and the user had to login only once for accessing two web applications, deployed on different servers.
The mechanism works only within the same browser window, since the user state is stored by the Identity Provider in session.
There are some javascript libraries (developed by Facebook) which allows cookie information to be transferred among browsers, so if the user is authenticated in one browser, than he is

ISSN 2067 – 4074

automatically authenticated in all other browsers that he might start.

4. Conclusions

SimpleSAMLphp is very useful for implementing Web SSO mechanism in web applications. It is developed in native PHP and supports integration to any SAML providers, PHP or non-PHP web applications. The library is very flexible, it ships with several authentication modules and further more can be easily adapted to third party applications.
In terms of Web SSO, the technology has become very popular especially with the rise of concepts like Web 2.0 and the continuous development of social networks websites like Facebook, MySpace and others.

References

[1] Single Sign-On http://www.opengroup.org/security/sso/

[2] Single Sign-On – A Contrarian View

http://www.databaseanswers.org/ibm_sso.htm

[3] Single Sign-On

http://en.wikipedia.org/wiki/Single_sign-on

[4] Build and implement a single sign-on solution

http://www.ibm.com/developerworks/web/library/wa-singlesign/

[5] Federated Identity

http://en.wikipedia.org/wiki/Federated_identity

[6] The basics of Identity Federation

http://www.sun.com/software/products/identity/basics_id_federation.pdf

[7] Security Assertion Markup Language

(SAML) V2.0 Technical Overview

http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

[8] Security and Privacy Considerations for the OASIS Security Assertion Markup

Language (SAML) V2.0

http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf

[9] Assertions and Protocols for the OASIS

Security Assertion Markup Language

(SAML) V2.0

http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[10] How to configure simpleSAMLphp 1.3 as SP and Shibboleth 2.1 as IdP

http://www.zeitoun.net/articles/configure-simplesaml-1.3-sp-and-shibboleth-2.1-idp/start

[11] Identity federation using SAML and WebSphere software

http://www.ibm.com/developerworks/webservices/library/ws-SAMLWAS/index.html?ca=drs-

[12] SimpleSAMLphp

http://simplesamlphp.org/

Appendix 1

1. Demo web application – sample page (index.php)

```
<?php
// include simpleSAMLphp classes
require_once ('../simpleSAMLphp/lib/_autoload.php');
// require authentication using the Service Provider "default-sp"
$as = new SimpleSAML_Auth_Simple('default-sp');
$as->requireAuth(); // the execution stops and the browser is redirected to
the SP page
// the user has logged in
// retrieving the user attributes
$attributes = $as->getAttributes();
// display the user attributes
print_r($attributes);
?>
```

2. Service Provider – authentication sources (config/authsources.php)

```
<?php
$config = array(
      …
      // An authentication source which can authenticate against both SAML
2.0
      // and Shibboleth 1.3 IdPs.
        'default-sp' => array(
                'saml:SP',
                // The entity ID of this SP.
                // Can be NULL/unset, in which case an entity ID is
generated based on the metadata URL.
                'entityID' => 'localhost',
                // The entity ID of the IdP this should SP should contact.
                // Can be NULL/unset, in which case the user will be shown
a list of available IdPs.
                'idp'               =>              null,              //
'http://172.16.1.11:8000/simplesaml/saml2/idp/metadata.php',
                // The URL to the discovery service.
                // Can be NULL/unset, in which case a builtin discovery
service will be used.
                'discoURL' => NULL,
        ),
…
```

3. Service Provider – Identity Provider metadata (metadata/saml20-idp-remote.php)

```
<?php
$metadata['http://172.16.1.11:8000/simplesaml/saml2/idp/metadata.php']    =
array(
    'SingleSignOnService'                                                =>
'http://172.16.1.11:8000/simplesaml/saml2/idp/SSOService.php',
    'SingleLogoutService'                                                =>
'http://172.16.1.11:8000/simplesaml/saml2/idp/SingleLogoutService.php',
    'certFingerprint'      => 'a1d10ecd56c3e6c17edeb527acbe1befa0a5ef88',
);
?>
```

4. Identity Provider – authentication sources (config/authsources.php)

```
<?php
$config = array(
```

ISSN 2067 – 4074

```
…
        'myauth' => array(
                //'saml:SP',
                'sqlauth:SQL',
                'dsn'                                                   =>
'mysql:host=localhost;port=3306;dbname=simplesaml',
                'username' => 'test_user',
                'password' => 'test_pass',
                'query' => 'SELECT `username`, `name`, `email` FROM `users`
WHERE `username` = :username AND `password` = :password',
        ),
…
```

5. Identity Provider – Service Provider metadata (metadata/saml20-remote-sp.php)

```
<?php
$metadata['localhost'] = array(
        'AssertionConsumerService'                                      =>
'http://localhost/simplesaml/module.php/saml/sp/saml2-acs.php/default-sp',
        'SingleLogoutService'                                           =>
'http://localhost/simplesaml/module.php/saml/spsaml2-logout.php/default-
sp',
);
```

6. Identity Provider – MySQL table to store the user credentials

```
DROP TABLE IF EXISTS `simplesaml`.`users`;
CREATE TABLE  `simplesaml`.`users` (
  `id_users` int(10) unsigned NOT NULL auto_increment,
  `username` varchar(255) NOT NULL,
  `name` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  PRIMARY KEY  (`id_users`)
);
```