# Web Single Sign-On Systems

Shakir James, scj1@cse.wustl.edu

## Abstract:
Currently, many web applications require users to register for a new account. With the proliferation of web applications, it has become impractical to expect users to remember different usernames and passwords for each application. Web Single Sign-On (Web SSO) protocols allow users to use a single username and password to access different applications. This paper examines three Web SSO protocols: SAML Web Browser SSO Profile, WS-Federation Passive Requestor Profile, and OpenID.

## Keywords:

## Table of Contents
1. Introduction
   1.1 Related Work
   1.2 Organization
2. SAML
   2.1 Core
   2.2 Bindings
   2.3 Profiles
3. SAML Web Browser SSO Profile
   3.1 Framework
   3.2 Security Issues
4. WS-Federation Passive Requestor Profile
   4.1 Framework
   4.2 Security Issues
5. OpenID
   5.1 Framework
   5.2 Security Issues
6. Summary

## 1. Introduction
Today the proliferation of web applications force users to remember multiple authentication credentials (usernames and passwords) for each application. Faced with the impractical task of remembering multiple credentials, users reuse the same passwords, pick weak passwords, or keep a list of all usernames and passwords. Managing multiple authentication credentials is annoying for users and weakens security for the authentication system. Web Single Sign-On (Web SSO) systems allow a single username and password to be used for different web applications. For the user, Web SSO systems help to create what is called a federated identity.

Federated identity management benefits both the user and the application provider. Users only remember one username and password, so they do not have to suffer from password-amnesia. Application providers also reduce their user management cost. They neither need to support a redundant registration process nor deal with one-time users creating many orphan accounts. Therefore, all stakeholders benefit from a Web SSO standard. Microsoft’s Passport protocol [Microsoft01] was the first attempt at creating a Web SSO standard, but the protocol was never widely deployed by non-Microsoft vendors. The fact that major security flaws in Passport were found by [Korman00] may have hindered its adoption.

Passport is not the only Web SSO protocol that had vulnerabilities, flaws in others protocols were identified by [Pitzmann03, Groiss03]. Although Web SSO is conceptually similar to Single Sign-On (SSO) protocols like Kerberos, they must operate within limitations of commercial browsers. Web SSO systems are proxy-based true SSO systems [Pashalidis03]. Web SSO protocols are also called browser-based or zero-footprint protocols since they operate within the constraints of web browsers. Furthermore, messages exchanged in bro web-based protocols must be encapsulated in Hypertext Transfer Protocol (HTTP) supported by all commercial browsers since users should not be expected to install software to support a new protocol. Moreover, browser cookies are tied to a single domain so cannot be used for the multi-domain SSO (MDSSO).

MDSSO refers to the case where SSO occurs between security domains operated by disparate organizations. For example, in the web-based MDSSO, Alice uses her browser to book her flight at airineinc.com and is transparently logged in at hoteline.com to reserve her room. A trust relationship must exist between the two domains to support this federated authentication since the domains may have different security policies. This paper does not cover the trust establishment problem. It also does not focus on the federation of attribute and authorization data. It deals with federated authentication of users in the web-based MDSSO case.

Security analysis of web-based MDSSO protocols demands unconventional requirements and models, and to date only one protocol has been proved cryptographically secure [Groiss03]. In the paper, we focus on the three-party authentication protocols for Web SSO since these protocols have been analyzed by [Groiss03, Groiss05]. The three parties include a service provider (SP), identity provider (IP), and user agent (UA). The UA is the web browser operated by the user, the IP authenticates the UA, and the SP is the web application.

## 1.1 Related Work
In contrast to MDSSO web-based systems, some Web SSO systems provide SSO within a single organization. They are called Web Initial Sign-On (WebISO) systems. The Internet2 WebISO Working Group [WebISO03] attempted to standardize such systems, but the project is currently inactive. WebISO systems typically provide a web-based component to an organization’s existing single sign-on infrastructure, which currently does not support web-based authentication. Organizations have direct control over their security policies, authentication procedures, and direct access to their user database. Thus, cross-domain issues such as an inking trust relationships are not a major concern in WebISO systems. Many universities have developed and deployed custom WebISO systems [WebISO03]. Also notable is another Internet2 project called Shibboleth [Shibboleth07]. Shibboleth handles both the web-based MDSSO and SSO within an organization, and implements SAML v1.1.

In addition to federation authentication, MDSSO federated identity management also involves attribute-based authorization, pseudo-identifiers for privacy, single logout, and IP discovery. Typically most Web SSO protocols address these problems in conjunction with federated authentication. A mechanism for establishing trust relationships between organizations is also covered by SSO protocols.

## 1.2 Organization

Web ISO Working Group [WebISO03] attempted to standardize such systems, but the project is currently inactive. WebISO systems typically provide a web-based component to an organization’s existing single sign-on infrastructure, which currently does not support web-based authentication. Organizations have direct control over their security policies, authentication procedures, and direct access to their user database. Thus, cross-domain issues such as trust between domains are not a major concern in WebISO systems. Many universities have developed and deployed custom WebISO systems [WebISO03]. Also notable is another Internet2 project called Shibboleth [Shibboleth07]. Shibboleth handles both the web-based MDSSO and SSO within an organization, and implements SAML v1.1.
2. SAML

The Security Services Technical Committee (STTC) of the Organization for Advancement of Structured Information Standards (OASIS) developed Security Assertion Markup Language (SAML). OASIS is a global organization that produces more web standards than any other organization [OASIS]. SAML is based on Extensible Markup Language (XML) and is used for communicating authentication and authorization data between web domains. It is arguably the de facto standard for federated identity management. Successful SAML implementations exist in industry, government, and academia [Wisniewski05]. Many identity management products support SAML. Moreover, other standards such as the Internet2 Shibboleth project, Liberty Alliance, OASIS Web Services Security (WS-Security), and eXtensible Access Control Markup Language (XACML) are based on SAML [Wisniewski05].

The modular design of the SAML framework allows its components to be combined to support a wide variety of deployment scenarios. SAML consists of core, bindings, and profiles/components. Figure 1 shows the relationship between the SAML components. The profile component describes the context in which SAML is used, and bindings specify the protocol used to encapsulate SAML messages. Bindings and profiles are based on the SAML core, which defines the format of messages and the generic request/response protocols.

2.1 Core

The core consists of security assertion that define the syntax and semantic of messages, and general request/response protocol for transferring assertions [Cantor05]. Assertions are XML packages that carry SAML statements about the user. For example, authentication assertion may contain statements saying how and when a user was authenticated. The request/response protocol is also specified in XML. In the protocol, a request is a query for an assertion and a response returns either the assertion or an error. For example, a request may be for a principal to be authenticated and the response will be an authentication assertion saying whether authentication was successful. The encapsulation of the SAML core, assertions and protocols, in another common underlying protocol is called a binding.

2.2 Bindings

SAML bindings specify how SAML protocol messages map to other common protocols such as Simple Object Access Protocol (SOAP) or HTTP [Cantor05]. Bindings use standard communications and messaging protocols to allow autonomous SAML-compliant systems to securely transfer messages. Either SAML or the underlying protocol supports mutual authentication, message integrity and confidentiality. For example, in the SOAP binding, either SOAP or SAML can be secured. XML signatures and encryption are used for application level security and TLS/SSL for transport layer security. The core and bindings define a SAML use-case called a profile.

2.3 Profiles

SAML profiles specify how SAML core and bindings are used within a specific application [Hughes06]. For example, the Web Browser SSO profile uses the Authentication request protocol and bindings for HTTP and SOAP. Many types of SAML profiles exist, but SSO specific profiles include the Web Browser SSO Profile, Enhanced Client or Proxy (ECP) Profile, Identity Provider Discovery Profile, Single Logout Profile, and Name Identifier Management Profile [Hughes05]. Although SAML defines many profiles, Web SSO was the primary reason for developing the standard [Lockhart06].
As a real world example, in Figure 3, we describe the deployment of the Web SSO profile using the HTTP Artifact binding. An artifact is a reference to a message. An artifact can be resolved to get the content of the message.

1. The UA attempts to access a resource at the SP.
2. To request SSO service at the IP, the SP issues a request artifact (a reference to the request message) to the ID using the UA as a middleman.
3. The IP asks the SP to resolve the request artifact.
4. The SP responds with a message containing the original request message [Cantor05a].
5. The ID authenticates the UA.
6. The ID sends a response artifact to the SP again using the UA as a middleman.
7. The SP asks the ID to resolve the response artifact.
8. The ID returns the content of the original response.
9. The SP grants or denies access to the resource based on the IP response.

For security, the specification recommends the artifact be transferred from the UA using a secure channel (SSL/TLS) and the SP and IP use source authentication before sending the contents of the original message.

### 3.2 Security Issues

A security analysis of the Browser/Artifact profile in SAML v1.1 was presented by [Groß03]. Groß identifies several vulnerabilities in the protocol including man-in-the-middle, message leakage, and replay attacks. He offers counter-measures, and recommends the use of “secure channels such as SSL 3.0 or TLS 1.0 with unilateral authentication” [Groß03]. Groß also notes that secure channels prevent man-in-the-middle and replay attacks, but not message leakage. Although he describes the protocol as the "most carefully designed browser-based protocols in federated identity management," he concludes that its security vulnerabilities make it unsuitable for use as an industry standard.

The SSTC responded to the [Groß03] in [SSTC05]. They incorporated some recommendations made by [Groß03] into SAML V2.0, and added counter-measures to threats [Groß03] identified. In general, the SSTC argues that vulnerabilities in underlying protocols, which SAML operates above, are not weaknesses in the SAML specification per se. They also point out that the SAML specification recommends using TLS secure channels, which counter two of the three attacks identified in [Groß03].

### 4. WS-Federation Passive Requestor Profile

Similar to SAML, the WS-* specifications is a modular architecture that aims to provide a flexible and extensible framework to solve the general web security problem [Lockhart06]. The Web Services Federation Language (WS-Federation) is part of the WS-* specifications that addresses the Web SSO problem for both browser-based applications and web services in particular [IBM02]. IBM and Microsoft and other companies developed the standard. The Passive Requestor profile targets browser-based SSO and other web-enabled devices, and the Active requestor profile focuses on SOAP-based web applications. Like SAML, WS-Federation can be combined with different transport and application layer protocols to
implement various Web SSO scenarios.

Also like the SAML Web Browser SSO profile, WS-Federation is based on different subcomponents. WS-Federation uses WS-Security framework, WS-Trust, and WS-SecurityPolicy [Goodner07]. WS-Security, standardized by OASIS, provides the mechanism for securing SOAP messages by using security tokens. WS-SecurityPolicy allows services to describe their security requirements, and WS-Trust defines the Security Token Service protocol for requesting/issuing security tokens. WS-Federation extends the Security Token Service (STS) model of WS-Trust.

4.1 Framework

WS-Federation builds on the WS-Trust STS model [Nadalin07] to provide, among other things, a Web SSO service. In the STS model, a security token is needed for parties to communicate. The STS assigns these security tokens [Lockhart06]. The UA first queries the SP to determine its security requirements before trying to access a resource [Goodner07]. The UA checks if it has a valid token using WS-SecurityPolicy expressions. If the UA does not have a valid token, it requests one from the STS. Since the STS is a web service, it also has its own security requirement. As a result, the UA must present a security token to the STS before requesting a token for the SP.

In the WS-Federation Passive Requestor profile, the STS plays the role of the IP authenticating the UA [Bajaj03]. The HTTP POST protocol is used to transfer security tokens. The message exchange pattern is shown in Figure 4.

1. The UA tries to access a resource at the SP.
2. Assuming the UA does not have a valid token, the UA is redirected to the IP.
3. The IP authenticates the UA by assigning it a valid token.
4. The UA presents the access token to the SP to gain access to the resource.

Figure 4: Model for achieving Web SSO with the WS-federation Passive Requestor profile.

4.2 Security Issues

[Groß05] analyzed the WS-federation Passive Requestor profile and proved it was cryptographically secure. Their work was significant because it was the first formal proof of any browser-based federation protocol [Groß05]. They showed formally that the profile provides authentication and successfully establishes a secure channel. In their proof, they created new models for user and browser, and they treated the secure SSL as a “blackbox submodule.”

Back to Table of Contents

5. OpenID

Unlike SAML and WS-Federation, OpenID is an open-source project with a community driven standardization process. Brad Fitzpatrick originally invented the protocol for his blogging application. OpenID Foundation whose mandate is to “enable and protect whatever is created by the community” now supports the community. [OpenID] Major industry vendors also support OpenID. For example, AOL is an IP or OpenID and Microsoft’s identity management software supports OpenID.

SAML and WS-Federation are flexible business-oriented specifications. They give implementers many options for customized deployments. On the other hand, OpenID is a user-centric, lightweight protocol. OpenID does not abstract away any details from its specification; it offers a direct solution to the Web SSO problem.

5.1 Framework

OpenID employs a decentralized architecture that takes advantage of the existing Domain Names Server (DNS) service. Anyone with a domain name can choose to be her own OpenID IP, users without their own domain name can register with companies they trust. OpenID effectively piggybacks on DNS to solve the IP discovery problem together with Web SSO. In the authentication protocol, users are assigned identifiers that are based on a domain name. For example, Alice has registered the OpenID alice@id-service.com with her IP. When she tries to access a protected resource at bob-blog.com she supplies her OpenID alice@id-service.com. Then bob-blog.com contacts id-service.com to authenticate Alice. To associate multiple OpenID identifiers to a single domain, OpenID uses the Yardis protocol [Miller06].

Figure 5 shows how OpenID without the Yardis protocol extension implements Web SSO.

1. The UA attempts to access a secured resource at the SP.
2. The SP requests the user’s OpenID.
3. The UA supplies the OpenID to the SP.
4. The SP determines the location of the IP based on the OpenID and sends an authorization request to the IP by redirecting the UA to the IP.
5. If necessary, the IP authenticates the UA.
6. The IP sends a signed authentication response to the SP via the UA.
7. The SP grants or denies access to the UA.

After (3), the SP can optionally establish a shared-secret key using Diffie-Hellman for signing and verification of messages.
5.2 Security Issues

Unlike SAML, the security of OpenID protocol has not been analyzed. What is more, the OpenID community does not present a document considering the security issues with the protocol. The specification documents mandates and recommends some security features such as the use of SSL/TLS, but a self-contained document is not available.

Phishing attacks are the major security issue in the OpenID protocol. In a typical phishing attack a malicious SP redirects a user to a fake IP that steals the user’s password. Another phishing attack involves a malicious SP masquerading as a trusted SP by hiding behind a redirect server. The OpenID specification mandates that the SP verifies a successful authentication message from an IP. Verification involves checking the domain name of the IP, ensuring the message is not a duplicate (by using nonces), and verifying the message signature. Other attacks such as man-in-middle and replay attacks are also possible but use of HTTP over SSL/TLS can mitigate these threats.

6. Summary

SAML and WS-Federation are both relatively mature standards. Major industrial players support them and many identity products support them [OASIS07, IBM07]. Both protocols have been independently analyzed and their specifications are well documented. One of these standards will arguably dominate the Web SSO arena.

On the other hand, OpenID has support from Microsoft and AOL, so it is a serious “standard”. And its apparent weaknesses in terms of maturity and formalism may be its strength. Being a lightweight protocol OpenID has a significantly lower barrier to entry than the other standards. Thus, many online projects are supporting OpenID. Also, the dynamic nature of the open-source community, specifically its less rigid model for standardization, may help the standard stay one step ahead of the other industry standards.

References


List of Acronyms

DNS  Domain Names Server
HTTP Hypertext Transfer Protocol
ID  Identity provider
MDSSO Multi-domain Single Sign-On
OASIS Organization for the Advancement of Structured Information Standards
SAML Security Assertion Markup Language
SOAP Simple Object Access Protocol
SP Service provider
SSL Secure Sockets Layer
SSO Single Sign-On
STTC Security Services Technical Committee of the Organization
STS Security Token Service
TLS Transport Layer Security
UA User agent
URI Unique resource identifier
WebISO Web Initial Single Sign-On
Web SSO Web Single Sign-On
WS-Federation Web Services Federation Language
WS-Security Web Services Security
XHTML Extensible Hypertext Mark-up Language
XML Extensible Markup Language

Back to Table of Contents