Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 4 (2012) 134 – 139

Procedia Technology

C3IT-2012

A Survey on Single Sign-On Techniques

V. Radha[a], D. Hitha Reddy[a]

[a]*Institute for Development and Research in Banking Technology, Road #1, Castle Hills, Masab Tank, Hyderabad – 500 067 (A.P), INDIA*

Abstract

Single sign-on (SSO) is a mechanism that uses a single action of authentication to permit an authorized user to access all related, but independent software systems or applications without being prompted to log in again at each of them during a particular session. It reduces the risk for the administrators to manage users centrally, increases user productivity by allowing mobility and allows users to access multiple services or applications after being authenticated just once. This doesn't mean that the SSO system unifies account information for all services, applications and systems, rather it hides such a multiplicity of account information into a single account that the user needs to login. Once the user login, the SSO system generates authentication information accepted by the various applications and systems. The concept of SSO can be used within an Intranet, Extranet or Internet. This report explores various methods of SSO and the advantages by adopting it. It also discusses on implementing various types of SSO and the protocols that are being used.

Keywords: Single Sign-On, OpenID Provider, Relying Party, Kerberos, SAML, OpenID, BrowserID.

1. Introduction

In present digital world, users have to access multiple systems for carrying out their day-to-day business activities. As the number of systems increase, the number of credentials for each user increases and thereby possibility of losing or forgetting them also increases. Single Sign-On can be used to solve many problems related to multiple credentials for different applications. Single Sign-on access to the main authentication centre enables users to get access to all other resources available. SSO helps to improve user and developer productivity by avoiding the user to remember multiple passwords and also reduce the amount of time the user spend on typing various passwords to login. SSO also simplifies the administration by managing single credentials instead of multiple credentials. It makes easy to manage the rights of a user arriving, changing function in or leaving the company, to quickly integrate added applications, delegate access rights during holidays without increasing the helpdesk's workload.

2. Types of SSO

The various types of SSO shown in Fig: 1, fall under different categories, based on where they are deployed (Intranet, Extranet, Internet); how they are deployed (architecture – Simple, Complex); the credentials they use (token, certificate..) and the protocols they use (Kerberos, SAML, OpenID..).Following picture shows the types of SSO and their classification:

2212-0173 © 2012 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.protcy.2012.05.019

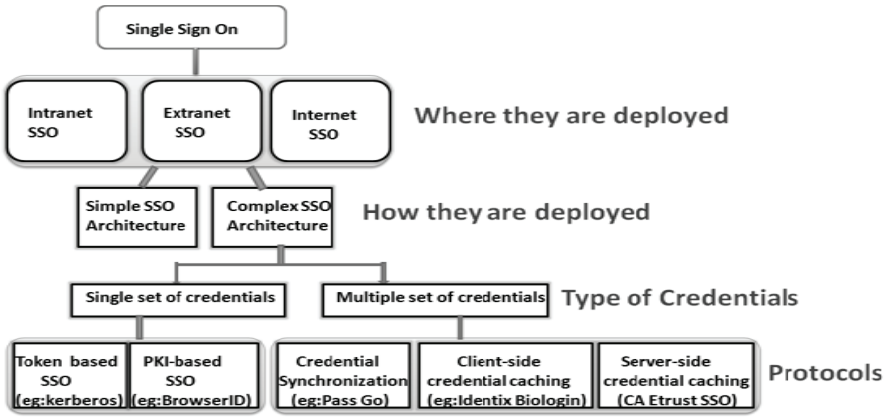

Fig 1: Classification of Single Sign-On

2.1. Where they are deployed:

2.1.1. Intranet or Enterprise SSO (ESSO):

Enterprise single sign on (ESSO) allows connecting to multiple systems within the same enterprise. ESSO is designed to minimize the number of times that a user must type their ID and password to sign into multiple applications. It automatically logs users in, and acts as a password filler where automatic login is not possible. Each desktop/laptop is given a token that handles the authentication.

2.1.2. Extranet or Multi-domain SSO:

Multi-domain SSO allows connecting to multiple systems within the same enterprise and all the business partners' applications. The user can login into one enterprise and access resources of the other, the users need not login again using different credentials.

2.1.3. Internet or Web SSO:

Web SSO is a browser-based mechanism, providing access with single login to applications deployed on web servers.

2.2. How they are deployed:

SSO architectures are divided based on their deployment as Simple SSO and Complex SSO as follows:

2.*2.1. Simple SSO architecture:*

Simple SSO makes use of single authentication authority, single set of credentials for each user. This architecture could be easily implemented in homogeneous LAN and intranet environment.

2.2.2. Complex SSO architecture:

Complex SSO uses multiple authentication authorities with single or multiple sets of credentials for each user.

2.3. Types of Credentials Used

Complex SSO can further be classified as two basic schemes, Complex SSO with a single set of credentials and Complex SSO with multiple sets of credentials.

2.3.1. Complex SSO with a single set of credentials:

Complex SSO using single set of credentials can be accomplished in two ways i.e. Token-based and Public Key-based as follows:

2.3.1.1. Token-based SSO system:

In this SSO system, a user submits the credentials to the token-based authentication authority, in which the credentials have been checked with its credential database. If the user credentials match, then the user is returned with a token. When the user wants to access an application server which is governed by second authentication authority, the same token is delivered to get a ticket to access the application server. Success of this process relies on the trust the authentication authorities have among themselves.

2.3.1.2. Token-based SSO in HTTP environment:

The Token-based SSO could be implemented by using cookies in HTTP environment. A cookie is a set of information given to the web browser by the web server and is stored in the client machine. Cookies used for authentication can be encrypted to keep them secret. The server could then retrieve the cookie and provide customized service to the client. Kerberos system provides basis for constructing secure SSO in network environment, however, it needs client side infrastructure and configuration. In HTTP-enabled environment, cookies could be used to construct SSO system and no extra installation or configuration is necessary. The biggest difference between Kerberos system and Cookies-enabled SSO system is that the former uses Remote Procedure Calls to transport authentication tickets, while the latter uses cookies to play the role of tokens.

2.3.1.3. PKI-based SSO system:

In PKI based SSO, the servers/resources and users authenticate each other by using their respective key pairs. Users can authenticate the servers by challenging the servers to decrypt any message they send which is encrypted by the public key of the server. Same way, servers can authenticate the user by challenging him to decrypt the message they send which is encrypted by the public key of the user. As the real owner of the private key only can decrypt, the mutual authentication i.e. server authenticating the user and vice-versa happens. The certifying authorities of users and servers can be different and if they are different there has to be trust among the certifying authorities.

2.3.2. Complex SSO with multiple sets of credentials:

2.3.2.1. Credential Synchronization:

The multiple sets of credentials needed to access multiple systems are masked by a single set of credentials to give an illusion that users need to remember only the single set. The synchronization software relieves the user from changing the credentials in all systems as and when the policy forces, by automatically forwarding the change request to all concerned authentication servers. eg: Pass Go

2.3.2.2. Client-side credential caching:

It allows users to store sensitive credentials like log on information (ex: user IDs and passwords) required for the websites or resources they access in a network. These credentials are stored in special folder called ***vaults***. With this stored information, user's system can automatically log on securely to the websites and the computers on their network automatically without requiring them to remember the

credentials all the time. Vaults can store all sorts of credentials like passwords, certificates, tokens etc. (eg: Windows Credential Manager).

2.3.2.3. Server-side credential caching: The Server-Side Credential Caching mechanism is same as Client-side Credential Caching architecture, with only difference being the credentials stored in a server instead of the client. It uses a central server to take on the task of administering all the different passwords and providing the needed information directly to the application asking for them. Eg: (CA Etrust SSO)

2.4. Single sign-on Protocols:

In this section we will discuss different protocols that are used on simple and complex SSO architectures.

2.4.1. Kerberos authentication Protocol:

Kerberos is a classical implementation of Token-based distributed authentication protocol. The whole process is divided into three parts among four entities. The four entities are 1) Client – the one who want to access resources 2) Authentication Server (AS) – the one who can authenticate the clients and resources 3) Ticket Granting Server (TGS) – the one who gives tickets to access resources and 4) Application Server (S) – a resource to whom the access is requested. The three processes are 1) Authentication Request and Response: in which the client using its credentials gets authenticated with AS and gets a key to securely communicate with TGS 2) Ticket Granting Request and Response: in which the client using the previously secured key from AS, requests TGS to get a ticket to access S and 3) Application Request and Response: in which the client uses the ticket it got from TGS to securely communicate with S. The first process where the credentials are required is completed by client only once and there after the 2nd and 3rd processes keep repeated as and when the client has to access other resources.

2.4.2. Security Assertion Markup Language:

Security Assertion Markup Language (SAML) is an XML-based open standard for exchanging authentication and authorization data between security domains, i.e., an identity provider and a service provider. Using SAML, an online service provider contacts an online identity provider which authenticates users who are trying to access secure content. SAML doesn't specify how to authenticate a user; rather it defines a way how to exchange the authentication and authorization data once the user is authenticated. SAML is nothing more than a series of XML-based messages called Assertions that detail whether users are authenticated (Authentication Assertion), what kind of rights, roles and access (Attribute Assertion) they have and how they can use data and resources (Authorisation Assertion) based on those rights and roles. It uses HTTP, SMTP, FTP and SOAP, among other protocols and technologies to transmit these assertions.

2.4.3. OpenID:

Open ID is a decentralized authentication protocol. OpenID consists of three main entities: 1) The OpenID Identifier: A String of text or an e-mail address that uniquely identifies the user; 2) The OpenID Relying Party (RP): A Web application or service provider that wants proof that the end user owns the said Identifier and 3) The OpenID Provider (OP): A central server that issues, stores and manages the OpenID identifiers of users. Relying Parties rely on this provider for an assertion that the end user owns the said Identifier. There are mainly four methods used in OpenID Protocol: 1.Discovery, 2.Authentication, 3.Association, 4. Verification.

Discovery:

End user initiates authentication by presenting a User-Supplied Identifier to the Relying Party via their browser. RP performs discovery (Discovery) on it and establishes the OP Endpoint URL which is used by user for authentication.

Authentication:

RP redirects the end user's browser to the OP with an OpenID Authentication request. OP establishes whether the end user is authorized. OP redirects the end user's browser back to the RP with either an assertion that authentication is approved or a message that authentication failed.

Association:

RP and OP establish an association with a shared secret established using Diffie-Hellman Key Exchange. OP uses this association to sign subsequent messages and RP to verify those messages; this removes the need for subsequent direct requests to verify the signature after each authentication request/response.

Verification:

RP verifies the information received from OP including checking the Return URL, verifying the discovered information, checking the nonce, and verifying the signature by using either the shared key established during the association or by sending a direct request to the OP.

4.4. BrowserID:

BrowserID is a decentralized identity system through which users can prove the claim of their email addresses allowing user's login into any website on the Internet using single password. It avoids site-specific usernames and passwords, an alternative for ad-hoc application level authentication. It implements Verified Email Protocol built by Mozilla, which offers streamlined experience. BrowserID consists of three main concepts: 1.Primary Authorities, 2.Relying Parties, 3. Secondary Authorities and the User Agent i.e. user's Browser.

- **Primary Identity Authorities (Primary):** A Service which provides the user with an identity in the form of an email address. It is an email provider like Yahoo! mail or gmail, builds BrowserID support.
- **Relying Parties (RPs):** Sites that use BrowserID for authentication.
- **The Implementation Provider (IP):** This is the user's web browser with native support for BrowserID, or else browserid.org serves web resources that implement the client portion of the system. It implements key management, required algorithms and serves as a Secondary Identity Authority.

BrowserID can be implemented by the following 3 steps:

1. **Certificate Provisioning:** Certificate Provisioning is the process in which a Primary verifies the user's email addresses and issues a signed certificate that proves user's ownership of that email
2. **Assertion Generation:** Assertion Generation is the process in which a user's browser produces an assertion that proves that a user owns given emails address.
3. **Assertion Verification:** Assertion Verification is the process in which a Relying Party can verify that an assertion of a user's ownership of a certain email is valid.

7. Conclusion

Single sign on undoubtedly makes it easier and safer by reducing to only one account per user for all services, number of passwords, central management of roles to define resources access control. It can be very beneficial to end-users, administrators and help desk. Single sign-on can gain much more importance with the emerging Cloud computing technology providing ICT services and also it reduces the chances of phishing attacks but as single sign on gives access with one login, it should be implemented in a secure way. Single sign on has its strengths and weaknesses and one must carefully estimate the use of the system and the resources available for its deployment and management before choosing SSO solution or else it can create a huge vulnerability in an organizations security if it's not implemented properly.

References

1. Elisa Bertino, Kenji Takahashi. Identity Management: Concepts, Technologies and systems. 685 Canton StreetNorwood,MA02062,ArtechHouse,2011;55-9,77-9,86-7,98-100,119. http://books.google.co.in/books?id=UrmD-Gxt-8IC&printsec=frontcover#v=onepage&q&f=false.
2. Colin Robbins, Edward Hamilton. Successfully deploying Single Sign-On (SSO) within an outsourced Environment. http://www.insight.co.uk/files/whitepapers/Single Sign on (White paper).pdf.
3. Jan De Clercq, Security Consultant HPCI Technology Leadership Group Hewlett-Packard. Single Sign-On Architectures. http://www.esat.kuleuven.be/cosic/seminars/slides/SSO.pdf.
4. Web Single Sign-On System for WRL Company. Si Xiong, Department of Internetworking, Royal Institute of Technology (KTH), Sweden; 2005; http://web.it.kth.se/~johanmon/theses/xiong.pdf.
5. Two SSO Architectures with a Single Set of Credentials. http://www.cs.auckland.ac.nz/courses/compsci725s2c/archive/termpapers/zlu.pdf.
6. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 OASIS Standard, OASIS; 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.
7. OpenID Foundation website. http://openid.net/.
8. BrowserID by Mozilla. https://browserid.org/.