Analysis of Single Sign-On Protocols from the Perspective of Architecture Deployment, Security and Usability

Abdullah Hussein Al-ghushami, Nur Haryani Zakaria, Norliza Katuk, Abubakar Mohammed
School of Computing
Universiti Utara Malaysia
06010 UUM Sintok Kedah, Malaysia
* Corresponding email: a_ghashami02@yahoo.com, haryani@uum.edu.my, k.norliza@uum.edu.my, abubakar33221@gmail.com

Abstract— Single Sign-On (SSO) requires one time authentication with a set of username and password which then allows an authorized user to enter all resources. This scheme was introduced to overcome the issue of memorability load among users who own several accounts. Currently, there are four main SSO protocols; 1) Security Assertion Markup Language (SAML), 2) OpenID, 3) InfoCard and 4) OAuth. These protocols were studied separately and they have different architecture deployment and implementation wise. It was found from the literature, that many users were not aware of the existence of those protocols which probably explain the slow adoption. Thus, this paper seeks to study the four protocols together by making further analysis and then compare them in terms of its architecture deployment and implementation wise focusing on security and usability perspective. It is much in hope that this paper will be beneficial in giving a better understanding of the SSO protocols, and contributes to better improvement in its implementation.

Keywords — Single Sign-On (SSO), Protocols, Architecture Deployment, Security, Usability

I. INTRODUCTION

Single Sign-On (SSO) is a type of technology where users are only required to authenticate themselves, one time only and then are given access to other security resources. This means that users do not have to re-authenticate again in order to get access to all other available resources on the internet, which saves time and effort. Furthermore, most of the users today own several accounts and they have to remember each and every single username and password which is very difficult to memorize all the passwords. Therefore SSO was introduced to solve the problem by allowing a user to use a single credential instead of multiple credentials to get access to multiple websites. SSO helps developers and users by escaping them to remember multiple passwords and also reduces the amount of time and effort that the users spend on entering numerous passwords to login. There are mainly four SSO protocols, namely Security Assertion Markup Language (SAML), OpenID, InfoCard and OAuth [1, 2, 3]. These protocols provide good credential management to the end users. Although, these protocols aim to provide users with single authentication, nevertheless its architecture deployment and implementation wise varies from one another. There are a number of existing studies [1,2,4,5] which focus on these protocols. Unfortunately, it was done separately, making it more challenging to view those protocols from various perspectives. In addition, the literature also suggests that the adoption of these protocols were not as rapid as it should be [6] which perhaps was due to the lack of overall understanding among the users on its implementation perspective. So by realizing this limitation, this paper seeks to study these four protocols by making further analysis and comparison of its architecture deployment in term of security and usability perspectives. This paper is organized as follows. Section II elaborates further on SSO architecture deployment. The analysis of the protocols from security and usability perspective is described in section III. Discussion in section V will entail the comparison between the protocols. This paper concludes in the final section VI along with possible future work to be undertaken.

II. SSO ARCHITECTURE DEPLOYEMENT

In order to discuss SSO architecture deployment, we classify the architecture into two; 1) Simple SSO architecture and 2) Complex SSO Architecture. The following sub-section will elaborate further on each type of architecture. Figure 1 below summarizes the two types of architecture deployment.

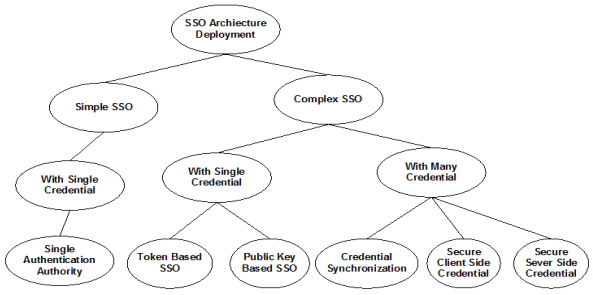

Fig 1. SSO Architecture Deployment

1. The Simple SSO Architecture

This architecture is simple to deploy in an authentication infrastructure which means that it can only be deployed with a single authentication authority. The users could hold a single set of credentials. Simple SSO could be easily deployed in homogeneous local area network (LAN) and intranet location. This is also approved by operating system

companies, especially when all computers run on a similar platform by trusting the same third party for authentication authority [2]. However, the simple SSO architecture could be complex when it expends for covering the diverse organizations and platforms that used diverse authentication credential and protocol govern by diverse authority.

2. The Complex SSO Architecture

The complex SSO architecture deployment on the other hand, governs many diverse authentication authorities with one or various sets of credentials for every user. This diversity in context means that SSO could be implemented on several platforms and covered by several organizations which use several authentication protocols and credentials [2]. Under the complex SSO architecture deployment it can further be classified according to either using single credential or many credentials.

2.1. With Single Credential

The architecture with a single credential used only one set of credential that can be acknowledged by several authentication authorities. Such as token based SSO and public key-based as follows:

2.1.1. Token Based SSO

Token based SSO is the simplest type by using only single set of credential. The users obtain temporary token whereby user token being validated by using cryptographic approaches which depend on secret key called symmetric cryptography [2]. The success of this process relies on the trust of the authentication authorities.

2.1.2. Public Key-Based SSO

In public key-based SSO architecture, each user requires to register his or her with trust authentication authority. It relies on private key and public key called asymmetric key. In contrast to token SSO architecture, the public key SSO requires updating based on latest technology [2].

2.2. With Several Credentials

This type of architecture is considered as complex due to deploying several different credentials and could be accomplished in three ways as follows:

2.2.1 Credential Synchronization

This type of architecture requires users to enter his or her credential in each single authentication. Therefore, with this, it is supposed not considered as SSO. However, there will be software that will help to synchronize the changing of a user's credentials in all applications. The policy will ensure that the change request to all concerned authentication servers is done automatically [2].

2.2.2. Secure Client Side Credential Cashing

This type of architecture involves a set of main credential being used to unlock for each user credential. This allows the user to access resources needed through diverse authentication credential. When the credential valid, it could be used to login to other resource servers whereby the secondary authentication depends on the primary one. The SSO system using secure side mechanism is not recommended to be deployed in the mobile devices (i.e, PDA) due to security concern since the architecture deployment is not making use of authentication infrastructure [2].

2.2.3. Secure Server Side Credential

In this architecture, the sever side store credential in the central repository. The set of credential used on the server side is not the same for each authentication authority. The server side offers good security due to all credential does not store in client disk [2].

III. ANALYSIS OF SIGNLE SIGN-ON (SSO) PROTOCOLS: SECURITY & USABLITY PERSPECTIVES

This section entails further analysis of the SSO protocols mentioned in the previous section. These protocols operate in federated environment whereby end users are allowed to login to the system at one time.

A. Security Assertion Markup Language (SAML)

SAML is mainly used in enterprises and schools/universities where the users will log on once and will be able to authenticate with other websites either internally or externally [7]. XML open source or standard is being used for developing the SAML. It is also used among identity provider (IdP) and Relaying Party (RP) for exchange authentication and authorization data. It defines rules and syntax for information exchange [8]. SAML also consist of XML messages called *assertions*. These assertions help to specify if users are being authenticated and what types of roles it deployed. There are several protocols such as SMTP, FTP, HTTP and SOAP are used to transfer these assertions. SAML is being used in some methods such as digital signature or encryption [7, 9].

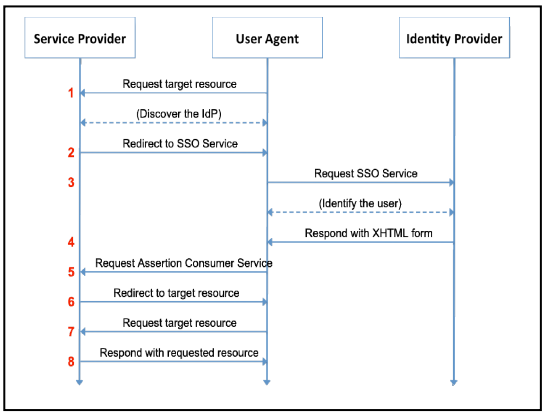

Fig 2. SAML Workflow [10]

B. OpenID Protocol

OpenID is a decentralized SSO protocols, which solves the problem of having an individual login and password for any website that support OpenID. It is an open protocol for any SSO systems [1]. It is user centric, which each user could be free to select his or her OpenID providers. Therefore, OpenID does not need any established trust among IdP and RP. OpenID Foundation [1], investigated billions of OpenID established user accounts which are offered by the

main IdP (e.g., Yahoo, Google, Facebook). This protocol allows the use of the available account for signing-in to several sites without creating another account with the site [11]. In this protocol, the identity of user's could be a URL, and OpenID authentication emphasizes that the user controls the content at that URL. The workflow of the OpenID protocol is shown in Figure 3.

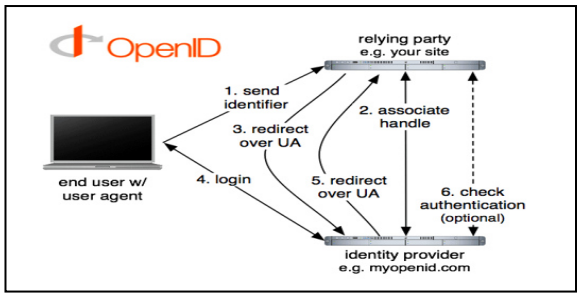

Fig 3. OpenID Workflow [12]

The good thing about OpenID is that it uses decentralized authentication, which indicate that there are no single IdP which users could select from. It depends on users and webs for selecting the third party, they would like to grant for the purpose of authentication. In comparison with other protocols, it is simpler, lighter and user friendly protocol. In other words, the focus of this protocol is more on scalability rather than security.

C. *InfoCard Protocol*

The Information Card protocol (called *InfoCard*) [4] being individual digital identity which is similar to a real identity card like a credit or license cards. Microsoft CardSpace [13] could be identity system which provides protection and consistent method for users in handling and managing personal information and in verifying identity for visiting sites. Figure 4 shows InfoCard workflow in how user login to RP website.

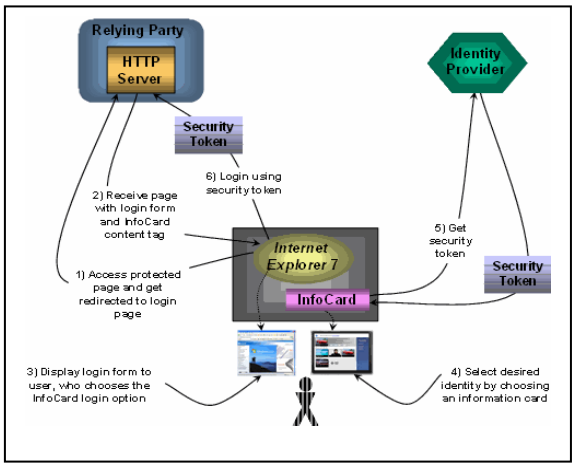

Fig 4. InfoCard Workflow [14]

It used to help sites or systems to get personal data from user sides. Every card is consisted assertions of a user identity which issued by either provider or users. When logging into a website, the user selects a card instead of his/her names and passwords. This card used to manage with the client side computer program (known as the identity selector (i.e.: Windows Card Space). Compared to other SSO protocols, InfoCard is considered as a heavy weight protocol due to users must install an identity selector. In addition, InfoCard is user-centered protocol for selecting security and privacy requirement and thus is more utilizable with other SSO application.

D. *OAuth Protocol*

OAuth is an open source for authorization mechanism and offer a generic framework to make the information owner authorize the third party from accessing owner data that save on server with not return to third-party credential like username and password [5,15]. Figure 5 shows how this protocol works.

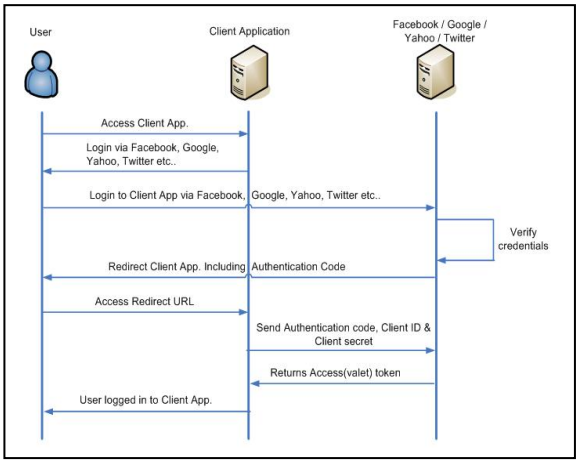

Fig 5. The Worklfow of OAuth Protocol [16]

It enables the user for sharing his or her resources as pictures and folders without relating to their identity data such as username placed on RP. It enables the consumer's application to access protected resources from RP by using an application program interface (API). It is utilized for securing API in portable and desktop systems [17]. There are two techniques of OAuth tokens namely 1) request token that used for authenticating user during cookie of its request via RP and 2) access tokens that used for obtaining requested information from RP [15].

IV. DISCUSSION

The previous sections have discussed about four main SSO protocols, and its workflow. Based on this, it is clearly shown that some of the protocols placed it's emphasized on security perspective while others may emphasize on usability. It is therefore becoming a big challenge for researchers to find the balance in-between security and usability. The following Table I summarize the comparisons made in between the four protocols based on the several characteristics, namely; purpose, implementation, techniques used, architecture deployment, security and usability.

TABLE I. SUMMARY OF COMPARISON BETWEEN SSO PROTOCOLS

Comparison	*SAML*	*OpenID*	*InfoCard*	*OAuth*
Purpose	Password store or enterprise users	existing account to sing multiple websites	Secure communicati on with other SSO	securing API in mobile application
Implemented between	IdP/RP	RP & IdP	identity selector/RP, IdP	RP and RP
Techniques used	SSO DS Encryption	URL & HTTP	Request and issue token	Request &access
Architecture Deployment	Complex sso(token)	Complex SSO	Complex SSO	complex SSO
Security	Secure (except for spoofed DNS)	Phishing/ Trust/ (DoS)	Phishing/mal ware /issues if used on pc shared	Secure but issue token stolen
Usability	Lack on usability	Used correctly if adopt&one OpenID login	Simple, but needed adoption.	used correctly/ hardly adopted

Today, there are several SSO architectures with diverse features and underlying infrastructures. Jan Clercq [2] specifies two core architectures to obtain SSO, including simple and complex SSO. The complex SSO deals with two solutions which are single and multiple sets of user credentials. The protocols discussed in this paper are mainly deployed on the complex SSO architecture [2]. Among the four protocols, many users prefer OpenID login. However, there are security concerns such as phishing and profiling, which need further thoughts [18]. The InfoCard protocol also has some security issues, particularly when the protocol is used on shared end devices (i.e. computer, laptop). The deployment of the protocols becomes difficult and complicated, especially when users switch among several workstations. This also affects the usability perspective which resulted in InfoCard face adoption issues among users. For example, the Microsoft did not continue in the implementation and development of InfoCard protocol since February 2011 [19]. On the other hand, OAuth protocol is considered secure when the whole end point from identity provider and relaying party were secure socket layer (SSL) protected.

From a usability perspective when deploying SAML, the users' login to any system has to rely upon a party which requires user identity. The IdP has made session of that particular user. The service provider has already obtained the identity of the user and the provider already has a session established for that user. In that case, there is no necessity to request for users' security credentials whereby the identity provider can just return the identity of the user to the service provider without bothering the user at all. Hence, it is becoming less troublesome for the users. Unfortunately, from a security perspective, DNS of users are spoofed (i.e. attackers are able to impersonate SAML IdP and can get user credentials) for later usage.

V. CONCLUSION

This paper discussed the comparison between those major protocols from various perspectives mainly on architecture deployment, workflow, security and usability. Based on the comparison made, this study found that each protocol is still prone to several security issues (e.g.: phishing, malware, spoofing) which requires specific attention to solve it. In terms of usability, OpenID and OAuth seems more user friendly compared to the others, nevertheless OAuth is hardly adopted due to the focus is more on authorization not authentication. It is much in hope that this paper will be beneficial in giving a better understanding of the SSO protocols, and contributes to better improvement in its implementation.

REFERENCES

[1] R. Tawil, D. Recordon and B. Fitzpatrick. OpenID authentication 2.0. http://openid.net/specs/ openid-authentication-2 0.html, December 2007.

[2] V. Radha and D. H. Reddy, "A Survey on Single Sign-On Techniques," *Procedia Technology*, vol. 4, pp. 134–139, Jan. 2012.

[3] A. Armando, R. Carbone, L. Compagna, J. Cuéllar, G. Pellegrino, and A. Sorniotti, "An authentication flaw in browser-based Single Sign-On protocols: Impact and remediations," *Computers & Security 33 (2013): 41-58.*

[4] A. Nanda and M. B. Jones. Identity Selector Interoperability Pro_le V1.5. http://informationcard.net/speci_cations, July 2008.

[5] D. Hardt, "The OAuth 2.0 authorization framework," *The Internet Eng Task Force RFC 6749*, October 2012.

[6] S. Feld and N. Pohlmann, "Security analysis of OpenID , followed by a reference implementation of an nPA-based OpenID provider." In *ISSE 2010 Securing Electronic Business Processes*. Vieweg+ Teubner, 2011. 13-25.

[7] K. D. Lewis, J. E. Lewis, and D. Ph, "Web Single Sign-On Authentication using SAML," vol. 2, pp. 41–48, 2009.

[8] Lewis, Kelly D., and James E. Lewis. "Web Single Sign-On Authentication using SAML." *International Journal of Computer Science Issues (IJCSI)* 7.4 (2010).

[9] Sun, San-Tsai, et al. "A billion keys, but few locks: the crisis of web single sign-on." *Proceedings of the 2010 workshop on New security paradigms*. ACM, 2010.

[10] "Logging Into SAML/Shibboleth Authenticated Server Using Phyton" (Stackoverflow) [online] 2013, http://stackoverflow.com/questions/16512965/logging-into-saml-shibboleth-authenticated-server-using-python (Accessed: 12 June 2015).

[11] B. Adida. EmID: Web authentication by email address. In Web 2.0 Security and Privacy Workshop 2008, Oakland, California, USA, 2008.

[12] "Using OpenID" (MediaWiki) [online] 2013, http://strattonbrazil.com/wiki/index.php?title=Using_OpenID (Accessed: 12 June 2015).

[13] Microsoft Corp. Windows CardSpace. http://www.microsoft.com/windows/products

[14] Chappell, D. "Introducing Windows CardSpace" (Microsft Developer Network) [online] 2006, https://msdn.microsoft.com/en-us/library/aa480189.aspx (Accessed: 12 June 2015).

[15] V. Jain and A. Sharma, "A Taxonomy on Cloud Computing," vol. 4, no. 3, pp. 149–153, 2014.

[16] "Typical OAuth 2.0 Workflow" (Tumb1r) [online] http://inerciatech.com/post/53930879442/this-is-the-typical-oauth-2-0-workflow-the-oauth (Accessed: 12 June 2015).

[17] F. Yang and S. Manoharan, "A security analysis of the OAuth protocol," *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pp. 271–276, Aug. 2013.

[18] C. Messina. OpenID Phishing Brainstorm. http://wiki.openid.net/OpenID Phishing Brainstorm, 2009.

[19] Al-sinani, H. S., & Mitchell, C. J. , *Using CardSpace as a Password-based Single Sign-on System*, 2011,. 0–23.