# An Analysis of Web Single Sign-On

Shane Wang
University of British Columbia
Electrical and Computer Engineering
2211 Wesbrook Mall
1-604-822-7932
shane.wang@ubc.ca

## ABSTRACT
This survey paper examines the current state of single sign-on solutions for the web. Results from a security analysis of three commonly found SSO solutions: Microsoft Passport/Live ID, OpenID 2.0 and SAML 2.0 will be discussed with a detailed outline of their authentication process as well as highlights of security issues with their implementations. Also discussed will be publicized security vulnerabilities for each of the three SSOs. Finally, the survey paper will explore two alternate SSOs: Simple Authentication for the Web (SAW) and Privacy Aware Identity Management (Web2ID) with similar discussions on their authentication process and security issues.

## Categories and Subject Descriptors
D.4.6 [**Security and Protection**]: Authentication

## General Terms
Security

## Keywords
Single Sign-on, Passport, OpenID, SAML, Web Single Sign-On, SSO, Authentication

## 1. INTRODUCTION
With the proliferation of web-based systems and applications, end-users are faced with the increasingly common issue of the need to memorize and keep many unique account usernames and passwords for each system or application. This poses an additional challenge to developers and support staff as many end-users invariably forget account details to less commonly used systems or many orphaned accounts are created and forgotten. To complicate the situation further, users commonly choose the same username and password for multiple accounts thus weakening the security for these authentication systems [2]. With authentication being a horizontal requirement that spans across all systems and applications, the requirement for unique and individualized accounts for each is unnecessary and overly complex.

Single sign-on protocols attempt to address this issue by allowing a user to use a single username and password to authenticate across multiple systems and applications. This is commonly accomplished by having an Identity Provider (IdP) that maintains user credentials which are then passed on to Relying Parties (RP) or Service Providers (SP) to authenticate users. This paper will provide a brief background on the history and current state of Web SSO and also an overview of the scope of the research paper. Three popular Web SSO implementations: Passport/Live ID, OpenID and SAML will be covered along with details of common issues and vulnerabilities. Finally, proposed solutions and alternatives will be covered with a review of proposed implementations.

## 2. BACKGROUND
Web SSO solutions were initially developed by various educational institutions in the mid-1990s. Such examples include Stanford University's WebAuth, Cornell University's SideCar, and Yale's Central Authentication System (CAS) were all early innovators in the field [1].

Commercial solutions did not emerge until Microsoft released their Passport, precursor to Windows Live ID, which was met with limited success as a centralized identity system [1]. Security issues with its implementation were later discovered [1] and its popularity amongst major systems waned in 2004.

Also developed during the same time period was the Security Assertion Markup Language (SAML). Established to be an open XML-based identity system where identities are federated where websites, also known as Service Providers (SP), relies on user authentication from another trusted party referred to as an Identity Provider (IdP). SAML continues to be a popular option amongst government and educational organizations today and is even being used in the Shibboleth framework implemented into the Campus Wide Login used by the University of British Columbia.

OpenID is another popular Web SSO that was developed as an open-source project with a community driven standardization process. SAML was often criticized as being too complex and enterprise centric in its design and implementation which in turn did not scale as well for internet wide use cases. These points of contention were all addressed with OpenID. Originally conceived by Brad Fitzpatrick for blogging, OpenID has since grown to be one of the biggest Web SSO contenders with support from major vendors such as Microsoft and AOL [1].

## 3. SCOPE
Many competing SSO solutions are currently deployed throughout the internet. SSOs such as Facebook Connect, Liberty Alliance, Shibboleth, CardSpace and OAuth all have had or continue to have a wide following. However, many of these solutions are not SSO centric as some delve into other topics such as identity federation (Liberty Alliance) and access control (OAuth) which are all considered out of scope for the study conducted in this paper. Also, solutions such as Liberty Alliance and Shibboleth are built with the same design principles as SAML so security issues and discussions of SAML will also be relevant to those two solutions. In this paper, the focus of study will be on Microsoft Passport/Live ID, SAML 2.0 and OpenID 2.0 which have all had a long history and gained a wide audience with many users and providers.

## 3.1 General Issues with SSO

Before investigation into the three more commonly found Web SSO solutions, the paper will first discuss the general issues commonly found in all Web SSO implementations.

Identified by Suriadi, "one of the main problems with the model is user privacy" [7]. In a SSO environment, relying parties (RP) or service providers (SP) "can also gather information about a user from the information they get from the identity providers (IdP). Sharing of user's information by malicious IdPs and SPs can reveal a complete user's identity and activities" [7]. This issue has caused users to be wary of SSO implementations and is one of the main reasons for the lack of wide spread adoption [21].

The concern of a single point of failure if the password to a SSO account is compromised provides an additional deterrence to widespread adoption [9]. Given the vulnerability to phishing attacks [14][15], RPs and SPs have been understandably cautious in adopting a standard solution. When a SP or RP builds their business model on the user's ability to authenticate and interact with their system, many business owners are reluctant to hand over the responsibility of authentication to a third party. By doing so, they are placing too much trust and reliance on the IdP and any security breach or disruption of service at the IdP could lead to potential disastrous loss of revenue or even legal liability. An inherent trust must be placed on the IdP by RPs or SPs to ensure all user information is secure and confidential [5]. RPs and SPs must assume liabilities associated with these breaches even if they occur due to improper or negligent practices on behalf of the IdP [6].

Finally, incentive for SP or RP to adopt a SSO system is also lacking. Most SPs or RPs "are reluctant to modify their login UI and process because new login procedures might confuse and upset existing users" [3]. Switching authentication mechanisms to a SSO solution means further education of the users is required and possible loss of user base if the transition is not smoothly executed. Exacerbating the situation is the lack of demand from users for a Web SSO solution. Studies have shown users are already satisfied by their own password managers [4].

## 3.2 Microsoft Passport/Windows Live ID

### *3.2.1 Implementation Details*

Microsoft Passport was the resulting product after Microsoft's purchase of Firefly Technologies in 1998 [22]. Originally intended as an authentication mechanism for its Hotmail email service, Microsoft later reintroduced Passport in 1999 as a fully featured SSO targeted at online shopping sites [24]. The service architecture involved one or more passport servers which served as Identity Providers that contained a list of registered users based on an email identifier and password. Also supported were mobile device registrations which contained their mobile number and a custom PIN (Personal Identification Number). All user accounts are assigned a 64-bit Passport User ID (PUID) for unique identification. In addition to the user account information, Passport also stored other details such as address, date of birth and credit card details [23]. Credit card details were only stored for use with Passport's wallet express purchase service. During account creation, users were able to select the information being shared with other SPs with the minimal shared identifier being just their PUID.

Service Providers opting to implement the Passport SSO solution must first sign a contractual agreement with Microsoft which requires a compliance testing fee of $1,500 and a $10,000 per annum provisioning fee [31]. Once accepted, the SP uses a PKI scheme with the Passport IdP to establish SSL/TLS channels for communication and authentication. The authentication protocol of Passport is as follows [23]:

1. User navigates to the participating SP site and clicks to login.
2. The client browser is directed to the Passport IdP server address which has been co-branded by the SP. The unique ID for the SP is sent to the Pasport IdP along with the SP site URL, usually as a query string.
3. Passport IdP examines the unique SP ID and attempts to match it to the return SP site URL to ensure only sites registered for the Passport service can use it for authentication. Passport IdP examines client browser's cookie cache for a "Ticket Granting Cookie" (TGC) which once decrypted allows the user to skip the rest of the authentication.
4. If no TGC cookie is found, the Passport IdP requests user to authenticate with credentials passed to the IdP via a HTTP Post method using SSL and TLS protocol. Once authenticated, the IdP places an encrypted, fresh version of the TGC cookie in the client browser's cache. Passport IdP also saves a set of cookies to the client browser's cookie cache with user's PUID and any other shared information for the SP to use.
5. Client browser is then redirected back to the SP which decrypts the encrypted information cookie and checks for authorization.
6. User is then authorized to access the resources at the participating SP site.

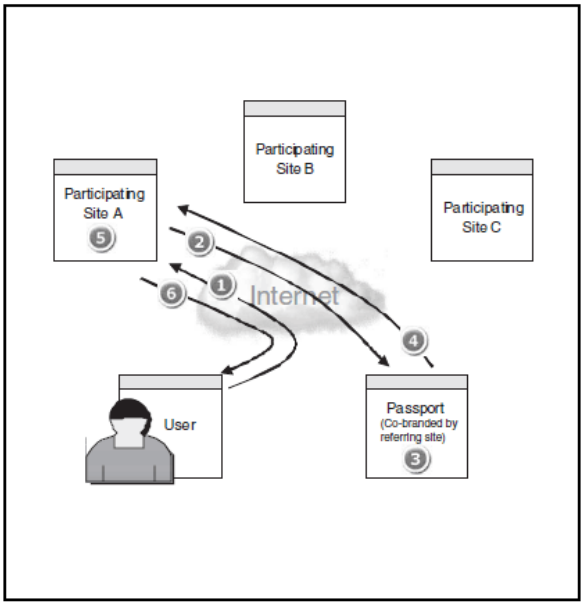


**Figure 1**: Passport Authentication Process [23]

In total, three encrypted cookies are placed on the client browser's cookie cache [23]:

1. Ticket cookie containing the PUID and time stamp (TGC).

2. Profile cookie containing user profile information the user has opted to share with SPs.
3. Visited Sites cookie containing a list of sites the user has signed in to.

All three cookies are encrypted using the Triple-DES (3DES) encryption algorithm through the use of a shared encryption key between the IdP and SP and are of type session cookies which are automatically cleared at the end of the browser session [24]. The encrypted cookies are transferred from SP to client browser via the query string of the URL. SP's have the additional option of ignoring valid ticket granting cookies and forcing the user to authenticate for additional security. However, Passport users do have the option of checking an option to "sign me in automatically" which is an option to save their credentials, including their password, to the computer [25].

**Table 1:** Stored User Information in Passport [25]

| Information | Data Type | Required to create a Passport? | Shared with other sites? |
|---|---|---|---|
| E-mail Address (Sign in name) | Credential | Yes | Only if user opts-in |
| Password | Credential | Yes | Never |
| Secret Questions and Answers | Credential | Optional | Never |
| Mobile Phone Number and Mobile PIN | Credential | Only required when registration takes place on a mobile device; otherwise they are optional | Never |
| Security Key | Credential | Optional | Never |
| Birth Date<br>Country / Region<br>First Name<br>Gender<br>Last Name<br>Occupation<br>Postal Code<br>Preferred Language<br>State<br>Time Zone | Profile | Optional, what fields are used is determined by the site that registers the user | Only if user opts-in |
| Billing addresses<br>Credit card information (Type, expiration date, etc.)<br>Name<br>Shipping addresses<br>Telephone number | Wallet | Optional, determined by the site that registers the user. | Only if user opts-in |

Once a user has explicitly chosen to sign out of Passport, the IdP examines the Visited Sites cookie to identify all SPs the user has signed into and redirects the browser to each SP to execute a script that signs out the user and removes the user cookie [25].

The Passport IdP has several mechanisms for preventing attacks. First, "if a user enters a password incorrectly five consecutive times, .NET Passport automatically blocks access to the account for two minutes" [25]. This prevents online dictionary or brute force password cracking attempts though "a determined and long-term brute-force attack still represents a potential threat" [25]. As previously mentioned, SPs requiring additional security has the option of ignoring valid TGC cookies and requiring the user to authenticate regardless of cookie validity. Since the authentication is done through a secure SSL/TLS channel, an attacker would therefore be unable to use a replay attack with hijacked cookies in this situation. Finally, for sites requiring even more stringent security, the Passport IdP provides the option of a two stage sign-in process where the first stage requires regular authentication and the second stage requiring a special four digit PIN entry. "A persistent failed-attempts counter for each user" [25] will track the number of failed entries and only reset once a user successfully logs in. "If a user fails five consecutive attempts, the system disables the user's security key" [25]. This method addresses the vulnerability of dictionary attacks present with just the normal stage one authentication scheme buts prevents an attacker from launching a DOS attack since they must possess the users regular sign in credentials.

### *3.2.2 Security Issues*

Since its deployment, Passport has been plagued with a series of highly publicized security flaws. Many of these details, which will be discussed, were openly revealed on the internet requiring Microsoft to make hastily patches. Even though the flaws were fixed, the damage to Microsoft's reputation was already done and many of the online merchants using the Passport service for online purchases moved away from the platform. Apart of the exodus from the service were eBay and Monster.com in 2004 as well as Expedia in 2009.

Also, the SSO received negative publicity from the Electronic Frontier Foundation's staff attorney, Deborah Pierce, for privacy issues relating to access of customer data in 2001 [26]. As a result, Microsoft moved to update their privacy terms to reflect proper usage and handling of customer data.

The first publicly revealed flaw of Passport was detailed in the paper "Risks of the Passport single signon protocol" by P.Kormann and A. Rubin [22]. First point of their discussion into the issues with Passport is the general usage of SSL. At the time of their paper, many browsers such as Netscape had 58 root public keys which all can issue certificates which are automatically trusted by browsers. Any of the 58 root public key holders that are less than vigilant in their certificate issuing could compromise the integrity of the system by issuing what a browser assumes to be a legitimate key to an attacker. Another key issue raised is the user interface of Passport and Hotmail. Hotmail, the email service, uses the Passport SSO for authentication. The interface for signing out of Hotmail and Passport were two distinct graphical user controls once a user has logged in. The user would presume that logging out of the Hotmail service by using the sign out control associated with Hotmail would remove their Passport credential cookies associated with the Hotmail service. Similarly, using the sign out control associated with Passport would remove not only their Hotmail service credentials but also all other Passport related services. The authors pointed out those less educated users would assume signing out of the Hotmail service would be enough for signing out. However, their Passport credentials still be logged in and any attacker with access to the computer would be able to access all services associated with the previous user's Passport account. Furthermore, it was discovered that the Netscape browser did not properly process a Passport sign out command. The authors discovered that after signing into Hotmail with Passport and then attempting to sign out from Passport did not remove all their locally stored Passport credential cookies. While the interface had indicated their sign out was successful and redirected them to the general MSN website, when they attempted to re-access their Hotmail account, they were able to do so without needing to re-authenticate. The authors then tried to repeat this procedure on various other machines with Netscape which all exhibited the same security flaw. Attempts to reproduce the issue with Microsoft's own Internet Explorer were unsuccessful. The issue was reported to Microsoft and promptly fixed though it revealed what appeared to be an issue in their internal testing process that did not cover other commonly used browsers. Finally, Kormann and Rubin pointed out that the

Passport SSO is highly susceptible to phishing attacks as a "bogus merchant threat is probably the weakest aspect of Passport" [22]. Setting up a fake web store front and getting signed SSL certs is easy enough. Creating a fake Passport authentication screen is also easily accomplished as an attacker can setup a misspelled domain such as www.pasport.com. Since users are so accustomed to co-branded SP logins, it is unlikely they will realize the site is not authentic and reveal their SSO Passport credentials which can then be sold or exploited by the malicious SP.

In the paper "Microsoft .NET Passport: A Security Analysis" [25], Oppliger discussed the security flaw discovered by a Pakistani computer researcher in 2003. Revealed on a full disclosure security mailing list, the researcher, Muhammad Faisal Fauf Danka was credited for the discovery. The flaw was associated with the .NET Passport's password recovery mechanism and allowed an attacker to reset any user's Passport account password to an arbitrary value through the use of their email address and an unpublicized web address with the phrase "emailpwdreset" [25]. It was theorized that this web address and its function was for internal Passport administrator use only. Once made public, Microsoft moved quickly to address this flaw though it is unknown how many users were affected by the issue.

Vulnerabilities to Cross Site Scripting attacks were also discovered. One method revealed by Slemko [29] allows an attacker to send a malicious email to a victim using Hotmail. The email contained an embedded iframe which contained a script to steal the logged in users Passport cookies. Once stolen, the attacker can use these cookies by copying them to their own browser cookie cache and as long as they have not expired yet, log into other SP. Also troubling is that logging into a merchant based SP site, an attacker can proceed to the shopping cart checkout where the Passport Wallet details such as credit card information where revealed to the user in a confirmation screen [28]. After this flaw was made public, Microsoft chose to retire the Wallet feature from its Passport SSO solution offering completely.

The most recent publicly revealed flaw with Passport (renamed at the time to Live ID) was related to a flaw that allowed new registrants to spoof email addresses associated with their Live ID account. "A critical error was made by the Microsoft programmers that allows everyone to create an ID for virtually any email address" [32]. As revealed by Duindam, a new user registering for a Live ID account can first input a valid email which the Live ID account confirmation is sent. Before confirming the account at the original email address, the Live ID account's registered email address can be changed to any arbitrary email address. Once changed, the user can use the original email confirmation to confirm the Live ID account, thus "the Microsoft system simply confirms the account, using the new and unowned email address" [32]. Attackers can then use in conjunction with social engineering to attack other users, tricking them into believing the communication is coming from another user, for example that works at Microsoft with an admin@microsoft.com email address. Although this security flaw has been fixed by Microsoft, it is unclear if previously spoofed accounts created through this flaw were also removed [33]. Without extensive auditing and logging, Microsoft would be unable to determine which accounts have spoofed email addresses putting into question the authenticity of the millions of current Passport/Live ID accounts in existence.

Given all the public flaws exposed and the steady rejection rate from large adopters of Passport, the SSO system is presently limited to supporting Microsoft's own service providers. As noted by Choo, "Microsoft Passport appears to be losing ground due to lack of trust, control and privacy; and the proliferation of other identity management paradigms" [31]. Businesses have opted out of the Passport SSO paradigm since the "idea of handing over sensitive data to a centralized "outsider" passport, may not be comfortable with individual organization" [31].

## 3.3 OpenID

As previously mentioned, OpenID began as a personal project of Brad Fitzpatrick, the creator of LiveJournal, in 2005 [37]. Started as a community standard for web SSO, it has grown and evolved into a robust solution, currently on version 2.0 of its specification, which is overseen and managed by a committee composed of both industry and community members [12]. The solution itself enjoys enterprise level support from a variety of web company giants such as Google, AOL, Microsoft, Verisign and Yahoo [37]. Even though the solution enjoys industry wide support, it is not without its flaws. Although documented, "the OpenID protocol is complex and only specified textually in a community standard document" [12]. The lack of details and ambiguities within its documentation has resulted in non-compliant and flawed implementations.

The "user-centric" architectural model of OpenID is distinctly unique from other SSO solutions in that the identity providers are unique and distributed. Users are free to choose from any number of IdPs which can then provide authentication credentials to relying parties or service providers. The protocol does not require any prior setup and establishment of a relationship between an OpenID IdP and RP or SP for authentication. The steps for an OpenID authentication is described below [11][12]:

1. Once a user has obtained an OpenID account from an OpenID Identity Provider, the OpenID IdP provides the user with an unique URL for his OpenID account (eg. https://openid.google.com/john). The user then attempts to login to a RP or SP that supports OpenID authentication.
2. The user then provides the RP/SP with his or her OpenID URL from his OpenID IdP which the RP/SP confirms by checking for the OpenID Extensible Resource Descriptor Sequence (XRDS) document.
3. Once verified, the RP/SP contacts the OpenID IdP to establish a shared secret for verification of messages received directly from the user.
4. The RP/SP then redirects the browser to the OpenID IdP's authentication page.
5. The user is notified of which RP/SP is attempting to authenticate to as well as a prompt to ask for permission to allow the IdP to accept authentication request from the RP/SP. Also passed along with the redirect is the identification of the RP/SP and the return URL once the user has been authenticated. The OpenID IdP can perform an optional RP/SP discovery at this open by examining the return URL and requesting a XRDS document from the RP/SP to confirm they are an OpenID RP/SP.
6. If the user had previously given permission for the RP/SP to receive authentication credentials from the OpenIdP and the user is already logged in, the user is simply redirected back to the RP/SP as authenticated.

7. Once authenticated, the browser is redirected back to the RP/SP where the RP/SP can directly confirm with the OpenID IdP authentication details and the RP/SP can gather specific information is gathered from the user (identity claim information).

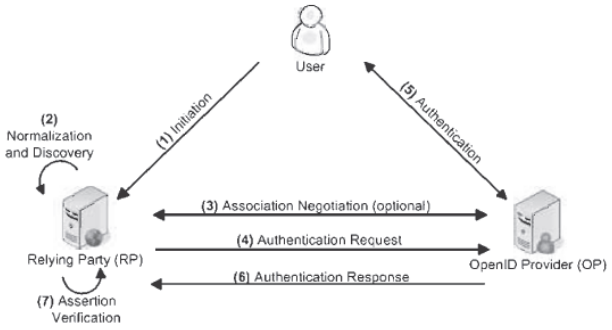


**Figure 2**: OpenID Authenication Process [11]

From a RP or SP standpoint, the OpenID SSO solution provides many of the benefits available from other standard SSO. The removed requirement of managing separate user registration process is welcomed by many. It also allows OpenID RPs/SPs to focus on obtaining only the necessary information from end users for their offering in the form of claims rather than the other repetitive but often unused personal profile details. This benefit allows the authentication and registration process to be much more streamlined and efficient from both the user and RP/SP perspectives. Additionally, users can also be guaranteed an additional level of privacy as the OpenID authentication scheme facilities the exposure of minimal amount of personal information while RP/SP do not have to worry as much about the accuracy of their registered user data.

OpenID is not without its drawbacks however. One big issue with OpenID and the distributed network of OpenID providers is that RPs/SPs are unable to provide a consistent look and feel throughout the authentication process. Due to the nature of multiple OpenID IdP, co-branding on the authentication page is no available. This shortcoming is one of the main reasons some large RP/SP such as Yahoo [34] has not adopted OpenID as an IdP due to the inherit phishing vulnerabilities.

### *3.3.1 Security Issues*

Phishing attacks are a great risk with the OpenID SSO solution [11][38]. Without the requirement of the RP/SP establishing a prior relationship with an OpenID provider for authentication, a malicious RP/SP can easily configure their authentication to redirect the user to their own OpenID IdP which can be designed to be visually identical to the legitimate IdP. As previous studies have shown, when phishing sites use visual deception, majority of users are unable to recognize the associated dangers and ignore most warnings and indicators [35]. "From a user perspective, the user is unlikely to be able to distinguish between an attack setup and a genuine SSO setup" [12]. After providing their OpenID credentials to the malicious IdP, the user is then redirected back to the RP/SP with no knowledge that their OpenID credentials have been compromised. This is a consequence of the architectural model where authentication can be done for any number of untrusted parties. Once compromised, a user's OpenID account can then be used by an attacker to access all the RP/SP sites the user belongs to.

Malicious OpenID IdPs are also another concern [12]. The malicious IdP can gather and store user credentials and associated RP/SP for a period of time before either using them or selling them causing both privacy and security related issues. This type of theft is both difficult to identify and detect [11]. Although a similar parallel can be drawn between OpenID IdP and email providers where the privacy and security concerns are equal in the presence of a malicious provider, users are much more educated and capable of identifying trustworthy email providers. Without a mature vetting process for OpenID IdPs, malicious IdPs will always remain a big concern.

Another common security concern is the exploiting of the redirect by malicious relying parties [12]. User credentials can be phished by a malicious relying party through the use of domain names that appear similar to the legitimate domain. Once a user visits this malicious site and attempts to login, the malicious site will pass along the return URL such that the legitimate RP/SP URL is used but a url redirect query string is embedded to return the user to the malicious RP/SP (eg. http://legitrp.com/redirect?url=http://1egitrp.com Note that 'l' has been replaced by '1'). In such a scenario, the OpenID IdP recognizes the return URL to be one of the trusted RP/SP the user has defined so the user is not asked for confirmation and can be automatically logged in. The actual return address however would be the malicious RP/SP address of http://1egitrp.com. Such an attack can be avoided by setting up the OpenID IdP to do RP discovery as described in previous step 5, prior to proceeding on to authentication. However, this is an optional step that not all OpenID IdPs implement due to the ambiguous protocol documentation for OpenID [12]. Review of the OpenID specification documentation revealed the use of the word "should" 48 times in describing recommended usage of the protocol [12]. Bellamy has identified that "many of these should be changed to must" [12] as providers of OpenID often take these to be optional recommendations which do not have any security related relevance to the implementation.

Denial of Service (DoS) attacks are also a great concern for RPs/SPs [11]. Using a large file or malicious script, an attacker can pass this along to the RP/SP when prompted for an OpenID IdP. RPs/SPs would then try to load the entire file or run the malicious script stressing their server. Countermeasures against this type of attack would require the RP/SP incorporate restrictions against the allowed protocols and ports for the OpenID identity as well as set specific data size limits [11].

Finally, depending on the implementation, OpenID is vulnerable to Man-In-the-Middle attacks [12]. TLS encryption is recommended along with RP/SP discovery to prevent such MITM attacks though not all OpenID instances use these. Without encrypted messages, adversaries are free to intercept and modify messages as they pass between OpenID IdPs and RP/SPs.

A recent study by Wang et. al [39] revealed a few instances of flaws and their exploits with OpenID. Through the use of a Browser Relay Messages (BRM) analyzer, it was determined that the message passed to the RP/SP for authenticating the user's identity contained non-signed elements that were writable by an adversary. In their demonstrated exploit, the authors were able to "cause the IdP to exclude the email element from the list of elements it signs, which will be sent back to the RP" [39]. Since the RP did not check that the email element was indeed signed by the IdP after receiving it, the authors were able to append another user's email to the message relayed from the browser and

impersonate and sign into the RP as another account. The flaw was eventually confirmed and patched by Google in their implementation of OpenID. A security advisory was also released shortly after from the OpenID Foundation acknowledging the issue.

An empricial anaylsis was done by Bellamy et. al. [12] where 32 OpenID relying parties were examined to see if they did implemented some of the recommended best practices of using RP discovery support and TLS encryption for both XRDS document and login. Results were less than ideal as only 8 sites incorporated all three recommended practices. "This seems to indicate a gap between how OpenID should be used and how it is used in practice. Whether or not an OpenID session is secure depends strongly on the implementation" [12].

**Table 2:** Vulnerability Statistics of 32 OpenID RP [12]

| Site rank range | Sites tested | RP discoverable | XRDS TLS protected | Login TLS protected |
|---|---|---|---|---|
| 1-1000 | 4 | 3 | 1 | 1 |
| 1001-10000 | 2 | 2 | 0 | 0 |
| 10001-50000 | 7 | 5 | 3 | 5 |
| 50001-100000 | 5 | 2 | 1 | 1 |
| 100001+ | 14 | 3 | 3 | 5 |

Analysis of OpenID IdPs raises similar concerns. In order to combat lax and insufficient OpenID IdP implmentations, the OpenID Provider Authentication Policy Extension was introduced in 2008 [36]. The PAPE allows OpenID RP/SP to specify to the OpenID IdP authentication policies to use such as multifactor authentication. Using the PAPE extension, OpenID IdPs will return authentication response indicating if the authentication policy specified by the RP/SP was met. The short coming of this method is that malicious OpenID IdPs as well as lazy IdPs will return false confirmation in an attempt to falsify the authentication scheme and have their authentication be used by as many RP/SPs as possible.

The best method of ensuring a secure environment is for OpenID IdPs and RPs/SPs to maintain and continuously update both a black list and white light for IdPs and RPs/SPs. RPs/SPs who do not implement the best practices such as TLS and discovery should be blacklisted from IdPs. Similarly, IdPs which do not meet PAPE standards set forth by RPs/SPs should also be black listed. Although this method is a viable solution in ensuring a more secure SSO infrastructure, it will not only require an additional effort in maintaining such lists but also go against the whole SSO paradigm, thus forcing users to choose multiple IdP accounts in some scenarios in order to access the full range of RPs/SPs they require.

Since OpenID is designed to be a "user-centric" solution, the ultimate responsibility of identity management and security is placed on the individual users. Users "should choose an OpenID provider which points out explicitly that the stored data is not used elsewhere" [11] and that the offered security features are in accord with their own security awareness" [11]. In addition, users can also adopt policies where multiple accounts can be created across various OpenID IdPs. For instances of RPs/SPs where account privacy and security is not as important, such as forums and chat sites, users can freely choose between any of the OpenID IdPs available. "These are also the kinds of accounts which users more frequently have to register for, and are a large source of password fatigue problems" [12]. However, for high value accounts where privacy is important and are related to monetary transactions such as Facebook and eBay, users can be vigilant and choose OpenID IdPs which utilize all the recommended best practices of implementation as well as ones that have a known reputation.

## 3.4 SAML

The Security Assertion Markup Language (SAML) is a SSO, XML encoded framework, currently on version 2.0, developed by the standardization organization OASIS. The protocol itself specifies the way security tokens, which contain assertions (or claims), about a user are passed from an Identity Provider to a Service Provider though the internet for SSO functionality. Version 2.0 of SAML is the resulting convergence of previous version 1.1 as well as the Identity Federation Framework 1.2 (ID-FF) from Liberty Alliance and Shibboleth 1.3 [40]. In addition, many large web companies such as Google have adopted the SAML protocol as the basis for their own SSO solution [18].

The SAML 2.0 protocol defines the various roles similar to previously discussed SSOs. Users have the ability to create one or more accounts across many SAML IdPs which can then be used across many RPs/SPs. Similar to the Microsoft Passport SSO, the SAML protocol requires that a prior trust relationship be established between an IdP and RP/SP. Once a trust relationship has been established, a user can then use their IdP to sign into the RP/SP. One important concept of the SAML protocol is the use of an artifact, or token, when communication occurs between parties [17]. The SAML artifact itself contains encoded data that provide assertions, or claims, about the user. The following describes the SAML 2.0 authentication process [16]:

1. User attempts to login to a RP/SP using a SAML 2.0 IdP.
2. The SAML IdP will authenticate the user and create the appropriate artifact and associated assertions. The IdP then redirects the user to the RP/SP with the artifact.
3. The user then is redirected and sends the artifact generated by the IdP to the RP/SP as an identification.
4. The RP/SP receiving the artifact from the user then sends a SAML:Request message to the IdP to verify the artifact and request desired assertions specific to the RP/SP.
5. The IdP receiving the SAML:Request message performs a lookup to ensure the RP/SP host matches the original URL of the delivered artifact created after step 1. Also, the IdP will perform a lookup for the artifact received from the RP/SP to ensure it has not been used before and responds using a SAML:Response message to the RP/SP with the requested assertions. All assertions are sent in a unique artifact.
6. Once the RP/SP receives the SAML:Response with the assertions, the RP/SP then determines if user is authorized to use their resources.

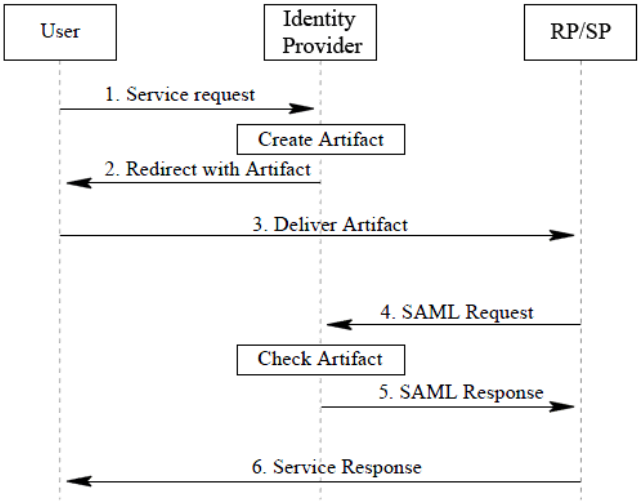


**Figure 3**: SAML 2.0 Authentication Process [17]

It should be noted that the specifications indicate that when the RP/SP sends the SAML request at step 4 of the authentication process, it only does so if the received artifact is parsable, which requires inclusion of the correct SAML artifacts and that the artifacts are valid (well formatted with correct version ID and contains same non-empty SourceID) [18]. Once the artifact has been checked, the established channel between the RP/SP and IdP is secure and bilaterally authenticated. Also, the SAML protocol specifies that the artifacts can only be used once and that RP/SPs do not store artifacts to avoid replay attacks. However, "if the transfer of the SAML artifacts to the source (IdP) site fails, the artifacts are still valid and reusable" [18].

Although implementations of the SAML 2.0 SSO protocol differs between systems, the overall security depends on two assumptions which are the trust relationships amongst the parties involved and the secure transport protocols used for message exchange [17]. The SAML specifications make many security recommendations which are important for avoiding the commonly associated pitfalls with SSO implementations. However, this does not mean implantations all follow these recommendations and that the associated pitfalls can all be avoided [16].

An important topic for SAML 2.0 implementations is the binding of the implementation. Binding refers to the way the SAML SSO will map assertions to transport protocols such as SOAP and HTTP. Communication between the user and RP/SP as well as the communication between the user and IdP should be established through SSL or TLS connections [18]. Under such conditions, communication between the user and other parties can be considered secure.

### *3.4.1 Security Issues*

First security issue raised by Grob [17] is associated with Man-in-the-Middle attacks for SAML IdP's not utilizing SSL/TLS encryption. "This lack of certification is a cornerstone of MITM attacks on the communication between the browser and source site" [17] whenever the browser first attempts to establish a connection with the SAML IdP for authentication. Since a MITM attack can be launched by an adversary at this point, "the two honest parties cannot distinguish the adversary from the intended communication partner" [17] from this point forth in the authentication process. Steps 1 and 2 in the SAML authentication process are both vulnerable to such an attack.

Replay attacks are also a possibility, especially during step 3 where the artifact is sent to the RP/SP [18]. Since the protocol does not "specify short-term freshness measures or the necessity of channel-based security, a replay attack may be possible" [18].

There are also some unique instances where security analysis has been done on a specific SAML based SSO solution. The study by Armando et. al [18] examines Google's SAML implementation for Google Apps. The authors analyzed the formal SAML 2.0 protocol by using a state-of-the-art model checker for security protocols (SATMC). With SATMC, the authors were able to determine "that two protocol sessions sharing the same IdP are sufficient for a malicious SP to mount this attack and gain access to a resource of another SP under the identity of an unaware user" [18]. The authors were able to reproduce the attack on the production deployment environment of Google Applications. For the experiment, ai-lab.it domain was registered at the Google's SAML SSO service and a corresponding public key for the IdP at the domain was given to Google. Through a Java Servlet, a dishonest SP called "BadSP" was simulated to construct SAML authentication requests which are forwarded to the AI-Lab IdP for authentication. Once a response is received, BadSP then recomposes a fake response for Google, thus allowing it to impersonate and successfully log into the Google Apps SP as the original AI-Labs member. Since publication of the leak, the authors have also reported the issue to Google who have promptly updated and patched their Google Applications service to address the vulnerability [18].

# 4. ALTERNATIVES

## 4.1 SIMPLE AUTHENTICATION FOR THE WEB

The first alternative SSO for review is suggested by Vander Horst and Seamons in "Simple Authentication for the Web" [20]. The suggested novel SSO method is similar in concept to automated email-based password reestablishment. Many current SSO solutions already utilize an email as the unique identifier for the user while their implementation adds an additional layer on top for authentication purposes. The authors argue since many RPs/SPs already utilize the email as a way to demonstrate ownership of an email address and perform an email based password recovery, "why not make email the primary means of authentication and remove site-specific passwords" [20]. Their suggested method of using email providers to authenticate users on behalf of RPs and SPs "works because web sites trust email providers to deliver messages to their intended recipients" [20]. This simplistic approach "removes the setup and management costs of passwords at sites" and "provides SSO without a specialized IdP" which thereby "thwarts all passive attacks" [20].

The authentication process in the suggested method is as follows [20]:

1. User attempts to login to a SP/RP by providing their email address.
2. The RP/SP generates a user token and an email token. The user token is given to the user and the email token is emailed to the user's email address.

3. User checks his/her email for the email token and once retrieved, sends it, along with the user token back to the RP/SP for authentication.

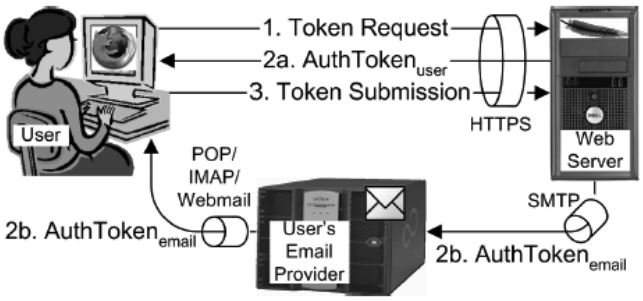


**Figure 4**: Simple Authentication for the Web Process [20]

For the authentication to be successful, the tokens must both be returned together to the RP/SP. This prevents eavesdroppers from intercepting just one message and attempting to replay it. Also, the tokens are single-use only removing further possibility of replay attacks.

There are obstacles present with such an implementation as the authors have noted. Latency issues with email delivery and lack of email encryption are two notable concerns. Users must ensure their email providers are sufficiently fast in message delivery, otherwise their authentication process will be delayed. Also, email providers should provide encryption of delivered messages to ensure the implementation is not vulnerable to passive eavesdropping and active modification. Also, for a user to manually check their email account for the email token is inconvenient and unnecessary. The authors suggested an automated process of retrieving email tokens, through web browser or email client extensions to improve usability of such a system [20]. Also another downside is that using the SAW method would not allow the user to pass additional assertions or claims to the RP/SP as many of the current SSO solutions do. Though the SAW method reduces password fatigue, registration fatigue whereby users are required to enter many of the same personal details across multiple RPs/SPs are still present.

The secondary authentication token is not limited to just email. As many RPs/SPs currently use an alternative method such as text messaging for password retrieval, the SAW method can be extended to use text or instant messaging as an alternative. The added benefit of this would be the lower latency associated with instant messaging [20].

## 4.2 PRIVACY-AWARE IDENTITY MANGEMENT

As mentioned earlier, one of the main issues with SSO is user privacy. Ensuring user information and activities are not shared amongst RPs or SPs is critical in the implementation of a secure SSO system. Zaradioon et. al. present an identity management protocol called Web2ID in "Privacy-aware Identity Management for Client-side Mashup Applications" [21]. The proposed protocol "describe a new relay framework in which communication between two SPs is mediated by a relay agent" which is shown to be privacy-preserving [21].

The proposed Web2ID "uses asymmetric cryptography to enable users to prove ownership of their identity URL without relying on any services by third parties" [21]. For the authentication, a user presents her credentials to the relay agent which in turn acts on behalf of the user and interacts with other RPs/SPs. The relay agent, also referred to as an identity mashlet, allows the user to perform authorization delegation and attribute exchange but only when explicitly defined by the user. Furthermore, the agent is designed so that an attribute requester can be anonymized, thus preventing the attribute provider from learning the identity and surfing habits of the user by knowing which other SPs the user has membership to. The authors noted that previous implementations using HTTP redirects to accomplish attribute exchange reveals the user's identity due to the need for the user to send a callback URL to the SP which in turn discloses their identity [21].

For identity establishment, a user first hosts their identity at an URL which contains the identity mashlet. At initial setup, the mashlet generates a public/private key pair with the public key embedded in the mashlet. The private key is kept and stored safely by the user. The following describes the authentication process for Web2ID:

1. User provides a RP/SP with his/her identity mashlet URL.
2. The RP/SP loads the identity at the supplied URL.
3. The RP/SP then locates the public key associated with the identity mashlet.
4. RP/SP then generates a session token encrypted with the user's identity mashlet's public key. Domain name is also included in the token to prevent replay attacks by adversaries.
5. RP/SP sends the encrypted token to the identity mashlet to decrypt.
6. The identity mashlet then asks the user to authenticate by providing the private key which then allows it to decrypt the token. Domain name of the RP/SP is then verified to the encrypted domain in the token.
7. Finally, the identity mashlet returns the session token back to the RP/SP to notify it of successful authentication.

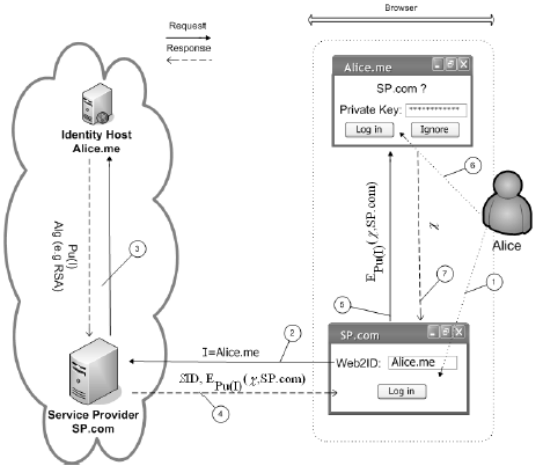


**Figure 5:** Web2ID Authentication Process [21]

The authors also state that their implementation uses MAC (Message Authentication Code) “to prove possession of session token” [21]. The RP/SP only serves requests that contain the correctly computed MAC value. An additional benefit of the protocol is that it is completely stateless and does not require user credentials to be transmitted over the network.

The implementation of the public key crypto allows the system to be protected from MiTM and other snooping attacks [21]. Also, considering that the domain name of the RP/SP is also embedded in the authentication token, replay attacks are difficult to launch.

## 5. CONCLUSION

With the ever evolving state of web services and applications, the need for a standardized SSO solution is more important than ever. From the analysis presented in this paper, it is apparent that there are still many issues which must be addressed before a web SSO system becomes widely deployed and adopted by providers and users.

Current SSO solutions are hampered by the use of existing technologies and protocols on top of which their implementations are built. However, due to the nature of the internet, SSOs must use these methods to ensure maximum compatibility amongst the various configurations of IdPs, RPs/SPs and users. The choice of choosing state-of-the-art and secure methods versus older but more compatible methods is a fine balancing act that all SSO solutions must take into consideration.

Enforcing SSO specifications and best practices is also a huge undertaking that is vital to the overall security of the network of IdPs and RPs/SPs. As noted in the studies of OpenID and SAML, specifications are not always followed by each and every IdP and RP/SP in the network. This eventually leads to security flaws and user privacy breaches that undermine the whole system. A system where IdPs and RPs/SPs are regularly tested and vetted to ensure proper implementation of the protocol specifications help tremendously in reducing the security vulnerabilities of such systems.

However, in its current state where many SSO solutions are vying for market share, password fatigue and many of the other issues SSO solutions are supposed to solve will still be present amongst internet users. Also, protocol specifications are more lenient on security implementations in order to obtain more widespread adoption. Unfortunately, with the fragmented and distributed model of the internet, it is doubtful there ever will be a single and robust SSO solution. In the meantime, the best approach for current SSO specification authors is to provide as much education as possible to implementers of their specification as well as continuously patch and improve upon their solution as new vulnerabilities are discovered.

## 6. REFERENCES


[1] Hodges, J., Howlett J., Johansson L., Morgan, RL. 2008. Towards Kerberizing Web Identity and Services. MIT Kerberos Consortium

[2] D. Florencio and C. Herley. A large-scale study of web password habits. In WWW '07: Proceedings of the 16th International Conference on World Wide Web, pp 657{666, New York, NY, USA, 2007. ACM.

[3] E. Sachs. Usability Research on Federated Login.http://sites.google.com/site/oauthgoog/UXFedLogin, October 2008.

[4] D. Mills. Identity in the Browser (Mozila Labs). https://mozillalabs.com/blog/2009/05/identity-in-the-browser/. 2010.

[5] A. Josang, M.A. Zomai, and S. Suriadi. Usability and privacy in identity management architectures. In the Proceedings of ACSW '07

[6] R. Dhamija and L. Dusseault. The seven flaws of identity management: Usability and security challenges. IEEE Security and Privacy, 6:24-29, 2008

[7] Suriadi, S., Foo, E., Jøsang, A. 2009. A user-centric federated single sign-on system. Journal of Network and Computer Applications 32.

[8] Heckle, R., Lutters, W. G., and Gurzick, D. 2008. Network authentication using single sign-on:the challenge of aligning mental models. In CHIMIT. ACM, Cambridge, Massachusetts.

[9] E. Maler and D. Reed. The venn of identity: Options and issues in federated identity management. IEEE Security and Privacy.

[10] T. Mustafić, A. Messerman, S.A. Camtepe, A. Schmidt, and S. Albayrak. 2011. Behavioral biometrics for persistent single sign-on. In Proceedings of the 7th ACM workshop on Digital identity management (DIM '11). ACM, New York, NY, USA, 73-82.

[11] Feld, S., and Pohlmann, N. Security analysis of OpenID, followed by a reference implementation of an nPA-based OpenID provider. In Information Security Solutions Europe (ISSE) conference (Madrid, Spain, 2008).

[12] Bellamy-McIntyre, J., Luterroth, C., Weber, G. OpenID and the Enterprise: A Model-Based Analysis of Single Sign-On Authentication. Enterprise Distributed Object Computing Conference (EDOC). 2011 15th IEEE International. pp.129-138, Aug. 29 2011-Sept. 2 2011

[13] H. Oh, S. Jin. The Security Limitations of SSO in OpenID. Advanced Communication Technology. 2008. ICACT 2008. 10th International Conference, vol.3, pp.1608-1611, 17-20 Feb. 2008

[14] H. Lee, I. Jeun, K. Chun, J. Song. A New Anti-phishing Method in OpenID. Emerging Security Information, Systems and Technologies, 2008. SECURWARE '08. Second International Conference. pp.243-247, 25-31 Aug. 2008

[15] Sun, S.-tsai, Hawkey, K., & Beznosov, K. (2010). OpenID Enabled Browser : Towards Fixing the Broken Web Single Sign-On Triangle. Computer Engineering, 49-58.

[16] S. M. Hansen, J. Skriver, and H. R. Nielson. Using static analysis to validate the SAML single sign-on protocol. In WITS '05: Proceedings of the 2005 workshop on Issues in the theory of security, pages 27{40, New York, NY, USA, 2005. ACM Press.

[17] T. Groß. Security analysis of the SAML single signon browser/artifact profile. In Proc. 19th Annual Computer Security Applications Conference, 2003.

[18] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. T. Abad. Formal Analysis of SAML 2.0 Web Browser Single

Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In 6th ACM Workshop on Formal Methods in Security Engineering (FMSE'08), pages 1–10, Hilton Alexandria Mark Center, Virginia, USA, 2008. ACM.

[19] W. Kaixing, Y. Xiaolin. A Model of Unite-Authentication Single Sign-On Based on SAML Underlying Web. Information and Computing Science, 2009. ICIC '09. Second International Conference on , vol.2, pp.211-213, 21-22 May 2009

[20] T.W. van der Horst and K.E. Seamons. 2007. Simple authentication for the web. In Proceedings of the 16th international conference on World Wide Web (WWW '07). ACM, New York, NY, USA, 1217-1218.

[21] S. Zarandioon, D. Yao, and V. Ganapathy. 2009. Privacy-aware identity management for client-side mashup applications. In Proceedings of the 5th ACM workshop on Digital identity management (DIM '09). ACM, New York, NY, USA, 21-30.

[22] D. P. Kormann and A. D. Rubin. 2000. Risks of the passport single signon protocol. In Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 51-58.

[23] P. McKiernan. 2002. Addressing Online Identity: Understanding the Microsoft Passport Service. In Information Security Technical Report, Vol 7, No. 3 (2002) 65-80.

[24] Rolf Oppliger. 2004. Microsoft .NET Passport and identity management. Information Security Technical Report, 9(1):26-34.

[25] Rolf Oppliger. 2003. Microsoft .NET Passport: A Security Analysis. Computer, Vol 36, No. 7 pp. 29-35.

[26] S. Olsen. 2001. Privacy terms revised for Microsoft Passport. CNet News. http://news.cnet.com/Privacy-terms-revised-for-Microsoft-Passport/2100-1023_3-255310.html.

[27] K. Regan. 2004. EBay Drops Passport in Blow to Microsoft. eCommerce Times. http://www.ecommercetimes.com/story/39325.html.

[28] B. McWilliams. 2001. Stealing MS Passport's Wallet. http://www.wired.com/science/discoveries/news/2001/11/48105.

[29] M. Slemko. 2001. Microsoft Passport to Trouble. http://www.znep.com/~marcs/passport/.

[30] Andreas Pashalidis and Chris J. Mitchell. 2003. A taxonomy of single sign-on systems. In Proceedings of the 8th Australasian conference on Information security and privacy (ACISP'03), Rei Safavi-Naini and Jennifer Seberry (Eds.). Springer-Verlag, Berlin, Heidelberg, 249-264.

[31] Kim-Kwang Raymond Choo. 2006. Issue report on business adoption of Microsoft Passport. Information

Management & Computer Security, Vol. 14 Iss: 3 pp. 218 – 234.

[32] E. Duindam. 2007. Windows Live ID Security Breached. http://www.exploring.nl/erik-duindam-us.pdf.

[33] J. Kirk. 2007. Microsoft Windows Live Flaw Opened Door to Scammers. About.com Computing Center. http://pcworld.about.net/od/instantmessaging1/Microsoft-Windows-Live-Flaw-Op.htm.

[34] A. Tom. 2010. What yahoo wants from Ops. OpenID Technology Summit West.

[35] Rachna Dhamija, J. D. Tygar, and Marti Hearst. 2006. Why phishing works. In Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI '06), ACM, New York, NY, USA, 581-590.

[36] D. Recordon, M. Jones, J. Bufu, J. Daugherty, and N. Sakimura. 2008. OpenID provider authentication policy extension 1.0.

[37] D. Recordon and D. Reed. 2006. OpenID 2.0: a platform for usercentric identity management. In Proceedings of the Second ACM Workshop on Digital Identity management, Alexandria, Virginia, USA, pp. 11–16.

[38] Drummond Reed, Les Chasen, and William Tan. 2008. OpenID identity discovery with XRI and XRDS. In Proceedings of the 7th symposium on Identity and trust on the Internet (IDtrust '08). ACM, New York, NY, USA, 19-25.

[39] R. Wang, S. Chen, X. Wang. 2012. Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services. In Proceedings of the IEEE Symposium on Security and Privacy. Oakland.

[40] S. Cantor. 2005. Shibboleth Architecture Protocols and Profiles. http://shibboleth.internet2.edu/docs/internet2-mace-shibboleth-arch-protocols-200509.pdf.