Security of Web Level User Identity Management

Jakov Krolo, Marin Šilić, and Siniša Srbljić
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
Phone: +385 1 6129 897
E-mail: jakov.krolo@fer.hr, marin.silic@fer.hr, sinisa.srbljic@fer.hr

Abstract - **The changing trends in the usage of contemporary Web technologies and Web design have led to the Web 2.0 concept. Web 2.0 has introduced variety of new possibilities for both Internet service providers and users. The rapid evolution of services like e-banking, e-commerce, social-networking sites, blogs, and video-sharing sites have arisen. The nature of these services requires for users to be digitally identified. The identification process is conducted on the Web services level and each service has its own user identity control system, which makes usage of services more difficult for users and raises development costs for service providers. In Web 2.0 era, instead of having the identity on the Web services level, identification process should be conducted on the Web level. This concept is known as Identity 2.0 and it represents a federated identity model in which users are in full control of their online identities. In this paper we discuss security risks of federated identity model. Furthermore, we review OpenID, the most popular protocol that implements federated identity model. Finally, we describe how OpenID responds to the security issues of federated identity model. As a potential solution to those problems, we discuss related protocols and interoperability between them.**

***Keywords*: Web security, Identity 2.0, federated identity model, OpenID**

I. INTRODUCTION

The recent Web design improvements and the way Web technology is utilized, and also the idea of using the Internet as a platform for application development, have led to the new Web 2.0 concept. The biggest turnover was the adoption of the AJAX that left the concept of static Web sites where users can view information to the new concept of building an interactive Web application in the Web browser. With the Web 2.0 concept, numerous new Web services with extended functionality have become available.

Contemporary Web services provide users not just the ability to browse the Web and access information, but also the ability to contribute and push their own content to the Internet. Web services like e-banking, e-commerce, social-networking sites, blogs, and video-sharing sites provide functionality that might have serious impact on reality and user activity can be reflected in real life. The nature of most Web services of that kind requires strict security settings management and the ability to reliably identify each user of a certain service.

Today, the existing Web services security management systems use centralized models or user identity management. Centralized identity management implies that each Web service embeds its own security management system for user identification, authentication and authorization. In a centralized model, security settings, access control and identity management are conducted on the Web service level. A centralized user identity management system requires users to register for service usage. During the registration process, users need to be assured that they will get their own unique user identity. A user identity management system usually assigns each user identification data consisting of a username and password. The main disadvantage is that the user needs to register and choose his identification data and, worst of all, the user needs to remember this data and provide it every time he wants to access the service. This approach to service access control and user identity management makes service usage more difficult for users. Another disadvantage of this approach to access control and identity management is that it significantly raises Web service development costs from the service provider perspective. In this way, every Web service that is deployed on the Internet needs to have its own access control management, registration management, authentication management, user identity data management and authorization management. An average Internet user has habits to choose the same username and password for different services, which leads to another approach to user identity.

Alternatively, instead of user identity management on the service level, significant improvement could be achieved when user identity management could be conducted on the Web level. This idea corresponds to the Web 2.0 concept and it is known as Identity 2.0. Identity 2.0 is a federated identity model that requires the assistance of a third party between users and services. The third party is responsible for user identification on the Web level. In this way, users would be identified on the Web level only once and then could access a variety of different services using the same identity.

The paper is structured as follows. Section II presents the federated identity model and security and privacy risks present in this model. In section III, we review the OpenID protocol, the most popular implementation of the federated identity model for user identity management. We show how OpenID manages security and privacy risks of the federated identity model. Section IV describes related protocols based on the federated identity model and their interoperability with OpenID. The paper finishes with conclusions in Section V.

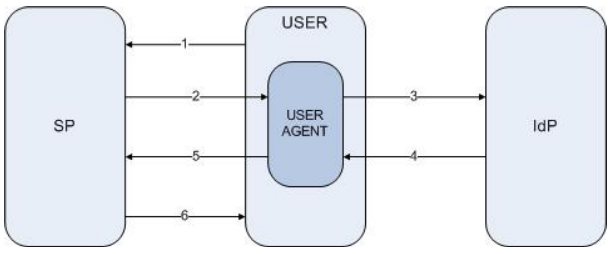

Figure 1. SSO SP-initiated pattern

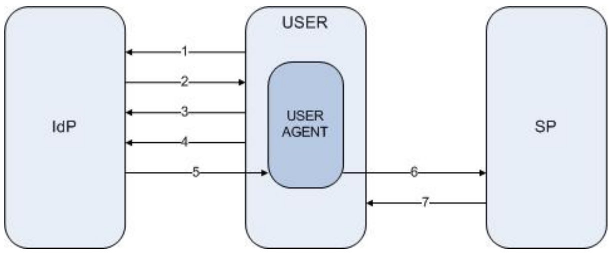

Figure 2. SSO IdP-initiated pattern

II. FEDERATED IDENTITY MODEL

The Federated identity model provides users the ability to distribute their digital identity across multiple security organizations and domains. The basic idea of federated identity management is to conduct user's identification on the whole Web level. This approach is called Single Sign On (SSO) [1] and it enables users to sign on only once and use the same identity to access multiple different Web services. In this way, using services and browsing the Web becomes easier and requires less effort. Another advantage is the reduction of development costs since the service provider offloads authentication to a third party.

There are four basic logical components in the federated identity model: user, user agent, service provider, and identity provider. The user is a person who acquires digital identity in order to interact with services in the Web environment. The user agent is a software application that runs on a PC or mobile device and is used bay the user for online interaction and Web browsing (usually a Web browser). The service provider (SP) is a Web application that exposes the user interface to the Web service functionality. The service provider offloads the authentication process to a third party and is also called the relying party in the federated identity model. The identity provider (IdP) is a Web application that conducts the identification and authentication process on the user's behalf on the Web level.

In the SSO approach, user identity data is transferred between the service providers and the identity provider. There are two basic variants of SSO from the data flow perspective. The first variant is called SP-initiated SSO [2] and the data flow in this pattern is presented in Figure 1. In the first step, the user tries to reach the service provider SP (1). The service provider receives a user request, generates an authentication request and sends it to the user agent (2). The user agent forwards the request to the identity provider IdP (3). IdP parses the authentication request, generates an encoded response and returns the response to the user agent (4). The user agent forwards the response to the service provider SP (5). The service provider parses the response and redirects the user agent to the originally requested service (6).

The other SSO variant is called IdP-initiated SSO [3] and its data flow is presented in Figure 2. In this pattern the identity provider is configured with specialized links that point to desired services. Initially, the user visits the identity provider IdP (1). IdP asks the user for identification data (2). The user provides his identification data to IdP and local security context about the user is created (3). Using the links on the identity provider site, the user requests the desired service provider SP (4). IdP creates authentication tokens from the local security context (5) and the user agent sends these authentication tokens to the service provider (6). The service provider checks if the user is authorized to access the service and redirects the user agent to the originally requested resource (7).

The SP-initiated variant of SSO is a form of user-centric identity management where everyone can implement their own identity provider. The main challenge in SP-initiated SSO arises in implementing an identity provider discovery from the service provider perspective. The most common answer to this challenge is to map identity providers with usernames patterns that users provide before the user agent is redirected to the identity provider. Security-wise, this pattern is more vulnerable to phishing attacks, since a malicious or compromised service provider could redirect the user to a fake identity provider that looks like the real one, and the user's identity could be compromised.

In the IdP-initiated pattern, the user visits an identity provider himself, and all the links that point to resources are embedded in the identity provider. The IdP pattern is more secure when against phishing attacks. However, the SP-initiated pattern is more scalable since the IdP-initiated pattern presumes that links that point to all the resources are available on the identity provider. In this way, every new service provider should register on each popular identity provider. This leads to identity management where few popular identity providers manage the majority of user's Web identities. If someone implements its own identity provider, then links that point to resources and services should also be embedded.

In the federated identity model, the user identification data source and identity management system are separated from its usage. The user can log in only once and access multiple resources on the Web without providing his personal data and identity to all of them. Service providers can focus on quality of service and core service functionality improvements. Identity providers can focus on identification and authentication methods improvement. Although all three parties involved in the federated identity model benefit from their participation, the model also introduces severe security and privacy issues. In the federated identity model, user identity is exchanged between the identity provider and the service providers. All parties should secure their communication channels against eavesdropping, man-in-the-middle attacks and other similar threats. In the HTTP protocol, the line is considered secure when SSL/TLS with mutual authentication is used. Another security issue is the authentication method that includes username-password pairs which is very vulnerable to phishing attacks. Since the identity is distributed across security domains, the risk of stolen identity is higher. Most of the protocols implementing SSO have a lifetime limit for security tokens. Finally, the role of identity providers in the federated identity model is to manage user identity on the Web level, which introduces privacy threat where a malicious identity provider could track user activity.

III. OpenID

The federated identity model is used as a basis for new identity standards that are being developed during the last few years. The most noticeable success was made by the OpenID standard. OpenID is an open decentralized standard for user identification, based on the federated identity model. According to statistics from January 2009 [4], more than 30 000 websites allow OpenID login as a means to access their services. The biggest Internet companies support or have announced plans to support OpenID. According to statistics from October 2008 [5], more than 500 million users are able to use OpenID authentication. Later on, Google [6], Microsoft [7] and PayPal [8] announced that they will start providing OpenID identities during 2009, significantly raising the number of OpenID users. We are approaching the point where a website can assume that the user has an OpenID identity.

A. What is OpenID

An OpenID identifier is usually in the form of a URL. For example, a user can sign on using the following OpenID identifier: "http://alice.example.com". Because URLs are unique, so are the digital identities represented by each URL. There is no connection between digital and physical identities, which means that a physical person can have more than one digital identity (just like he can have more than one e-mail account). It also means that one digital identity identifies exactly one Internet user. Besides URLs, OpenID identifiers can be represented by i-names, which are one form of the XRI standard. XRI is an open standard for sharing resources and data across domains and applications. It uses a new layer of abstract addressing over the existing IP numbering and DNS naming layers. XRI is intended to be as easy as possible for people to remember and use. For example, a user can sign on using the following OpenID identifier "=Alice", which is more likely to be used by an average Internet user than a URL.

Besides using URLs and XRI, it is OpenID's decentralized nature and cost advantage that made it so popular and accepted. OpenID is fully decentralized because users can host their own identity on any server they choose or have it hosted by one of many OpenID providers. OpenID providers can choose from a variety of software implementations from a variety of vendors and Open Source projects. OpenID does not crumble if any OpenID provider turns evil or goes out of business. What is important is that anyone can use their own technical innovations within the OpenID framework. This means that if someone decides he does not like the Diffie-Hellman cryptographic key exchange at the root of OpenID authentication, he can develop his own way of authenticating (e.g. using biometrics), and deploy it within the OpenID framework.

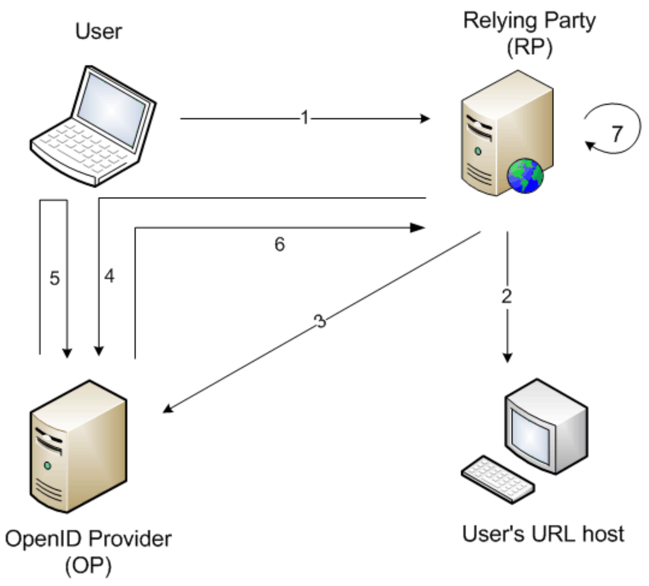

Figure 3. OpenID protocol

Besides the decentralization on many levels, OpenID's advantage is that OpenID's cost structure is fundamentally lower than having access-control systems for every website on the Internet.

B. Protocol overview

OpenID enables the user to keep control over his own identity by separating service providers and identity providers. The user registers his identity or account at a single identity provider, also called an OpenID provider (OP). With OpenID identity, the user has instant access to a vast number of service providers, also called the relying party (RP). The OpenID authentication protocol is defined in [9]. The following steps, as shown in Figure 3, are somewhat simplified.

Initially, the user visits the relying party RP and provides his OpenID identifier (1). Based on the OpenID identifier, RP performs discovery of the OpenID authentication service URL (2). RP and the OpenID provider OP establish an association. This step is optional, but recommended in order to establish a shared secret using Diffie-Hellman key exchange. OP uses an association to sign subsequent messages, and RP uses it to verify those messages. If this step is omitted, relying party would need to send subsequent direct requests to verify the signature after each authentication response from OP (3). RP forms an OpenID authentication request and redirects the user agent to OP (4). OP establishes whether the user is authorized to perform OpenID authentication and if he even wishes to authenticate (in case a fraud attempt is going on). The way the user authenticates to their OP and any policies surrounding such authentication is not part of the standard and is left to OP to decide on its own how it wants to implement it (5). OP redirects the user agent back to RP with authentication response, whether the authentication is approved or failed (6). 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 OP (7).

C. Security aspects

We discuss the most common attacks possible on the OpenID protocol. We talk about Denial-of-Service (DoS) attack, replay attack, man-in-the-middle attack, phishing, Cross-Site Request Forgery (CSRF) and privacy issues.

In the *DoS attack*, attacker tries to make a computer resource unavailable to its intended users. The relying party can suffer from a DoS attack if it allows a user to put any URL he wants as an identifier. A user could use an URL to some large movie (e.g. http://www.example.com/gigabytemovie.flv) and during the discovery process, the relying party would download the content that resides on that URL. To prevent this kind of attack, the relying party should limit the amount of data and time that can be consumed per request. An OpenID provider can also suffer from a DoS attack. This can happen if the relying party starts sending a large amount of requests for association, authentication or signature verification. To prevent this, an OpenID provider can use simple IP based rate-limiting. Also, OpenID can ban requests based on the values "openid.realm" and "openid.return_to" in protocol messages exchanged with the relying party.

Replay attack is an attack where an eavesdropper gets the information without authorization and then retransmits it to trick the receiver into unauthorized operations such as false authentication. According to OpenID specifications, the nonce, which stands for *number used once*, used in the authentication process does not have to be signed and verified. These kinds of authentication details are left for implementers to decide. If the nonce is not part of the signed request information and later verified, an eavesdropper could intercept a successful authentication assertion (sent from the OpenID provider to the relying party) and re-use it. To prevent the replay attack, the nonce should be part of the signed information in the request message sent from the relying party to the OpenID provider (the nonce value should be in "openid.sig" and "openid.signed" fields of the protocol messages) [10]. Also, when receiving the authentication assertion, the relying party should check if the nonce is correct. Another solution could be using the transport layer encryption (TLS) to prevent eavesdropping.

In the *man-in-the-middle attack*, the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection when in fact the entire conversation is controlled by the attacker. To prevent the man-in–the-middle attack, the protocol provides associations which prevent tampering of signed fields. Associations provide a shared secret between the relying party and the OpenID provider. Altering signed fields without knowing the shared secret requires breaking the MAC, but there is no known attack on the MAC used in OpenID. However, associations do not stop man-in-the-middle attacks in authentication steps 2 and 3 from Figure 3, i.e. during discovery and association sessions. OpenID depends on the URL, which means it depends on DNS, which is known to have security weaknesses. In case DNS is compromised, the attacker can impersonate the OpenID provider and issue its own associations. If an attacker can tamper with the discovery process, he does not even need to impersonate the OpenID provider, and can specify any OpenID provider. Additionally, if an attacker can compromise the integrity of the information returned during the discovery process, by altering the XRDS document, the need for a man-in-the-middle is removed. In that case, an attack can be prevented by digitally signing the XRDS file [11]. To prevent man-in-the-middle attacks, SSL with certificates signed by a trusted authority should be used for all parts of the interaction. This way the results of the DNS look-up can be verified against the certificate. Once the validity of the certificate has been established, tampering is not possible.

Phishing is a process of attempting to acquire sensitive information such as passwords by masquerading as a trustworthy entity. When a user enters his OpenID identifier on the relying party's authentication form, a malicious relying party can redirect him to a fake OpenID provider's website which looks pretty much the same as the authentic OpenID provider. Most users would not notice that they are being tricked and, by entering their password, would easily give away their credentials. This is the most classic form of phishing. Most users do not know or/and do not care about security, they just want to use the service. To prevent phishing, the most important thing is to get rid of the step where users type in the password. Probably the best solution would be using OpenID with Information Cards as a way of authenticating. Because Information Cards generate site-specific sign-in information and the attacker’s site is different than the authentic site, even when the user is tricked into submitting an Information Card to the evil site, the attacker does not have the ability to log into the real site. No shared secret was present to steal and no session was established to hijack. The type of authentication is not defined with the OpenID protocol. Whether OpenID providers will implement password-based authentication or Information Card based authentication is up to the OpenID provider. One other solution to phishing is using multi-factor authentication. Multi-factor authentication uses at least two kinds of authenticity verification, e.g. passwords plus phone/SMS verification. Using something you know (e.g. password) with something you own (e.g. phone) is a great phishing-resistant authentication method.

CSRF is an attack which forces an end user to execute unwanted actions (of the attacker's choosing) on a web application in which he is currently authenticated. The problem with OpenID is that by logging in, along with the username, the user provides the relying party the information that he is currently logged in to the OpenID provider OP. Malicious relying parties could use this information to submit forms to the OP utilizing the user's cookie, without the user knowing about it. This can be done using JavaScript to submit a hidden form in an i-frame, or similar. To protect against *CSRF*, OpenID provider must make sure that the form was served for the user and not for an attacker. The best solution is to put a hidden element into the authentication form containing a token based on a secret and data in the user's session object. This means that only a form served for a particular user will generate a submission valid for that user. Many reputable OpenID providers already use this method to secure form submissions and protect against CSRF.

The last security issue related to OpenID is a *privacy issue*. The OpenID provider can spy on its users by recording their Internet activity. The OpenID provider is authenticating the user for every relying party the user wishes to log into. Thus the OpenID provider can trace what relying parties the user is using. For better privacy, some kind of trust model should be introduced.

The OpenID standard does not define what authentication methods should be used. This way, it allows OpenID providers a market where each of them offers their own level of authentication strength. In December 2008, Provider Authentication Policy Extension (PAPE) [12] was introduced as an extension to the OpenID Authentication protocol. This extension provides a mechanism by which a relying party can request that particular authentication policies be applied by the OpenID provider in the authentication process. The PAPE extension also provides a mechanism by which an OpenID provider may inform a relying party which authentication policies were used. Thus a relying party can request that the user authenticate, for example, using a phishing-resistant or multi-factor authentication method.

IV. RELATED PROTOCOLS AND INTEROPERABILITY

The two most popular related standards based on the federated identity model are SAML (Security Assertion Markup Language) and InfoCard [1]. SAML is an XML standard for exchanging authentication and authorization data between security domains.

SAML is architected for security and privacy, and serves needs where strong requirements for trust and high-value transaction are needed. The protocol is composed out of assertions - XML packets containing user identity, authentication status and attributes. SAML removes security and privacy risks but its main disadvantage is difficult identity provider discovery in the general case. However, SAML is often used in a large trusted community. In this case, administrators configure service providers in that environment to contain the information about the identity provider. Another example of identity provider discovery with the SAML protocol is when a user can select his identity provider from the list of identity providers accepted by that particular service provider. OpenID and SAML both enable SSO and provide the ability of direct interactions between the identity provider and service providers, although not in the same way. The difference between OpenID and SAML is that OpenID is a lighter, service provider friendly and user-centric protocol more concerned with scalability than security [13]. On the other hand, SAML is a complex and heavy protocol, identity provider friendly and enterprise-centric, focused on security and privacy.

InfoCard [14] is a standard used by Windows CardSpace, the Microsoft .NET tool designed to provide users consistent digital identity. Windows CardSpace consists of collections of user data called identity cards. Each card represents different identity. User visits the service provider and chooses appropriate identity card when he is asked by the user agent. There are two types of cards, self-asserted cards and identity provider managed cards. Self-asserted cards are created by the user himself and stored directly on the user's device. Managed cards are issued by identity providers, who govern with user identity data. With managed cards, data is retrieved each time user selects these identity cards. The process of identity provider discovery is easily resolved for both types of cards. For self-asserted cards a user's device (e.g. his local computer) could be the identity provider. For managed cards, the identity provider stores its address on the identity card. InfoCard prevents an identity provider from knowing which service providers are served, which solves privacy issues related to identity provider spying. Also, InfoCard, like OpenID, uses user-centric identity management because the user has to choose the identity card and approve each authentication request. There is no direct communication between service providers and identity provider, as well as classical password input forms, which makes InfoCard phishing-resistant. Disadvantages of InfoCard are that it is not an open standard and it relies only on WS-* standards. However, interoperability of InfoCard and OpenID is possible. Integrating InfoCard into OpenID as a way of strong authentication would solve some important security issues that are not strictly addressed by the OpenID standard itself.

V. CONCLUSION

In this paper we described the concept of federated identity as a response to new security challenges and demands introduced with the Web evolution in recent years. We reviewed the most popular standard based on the federated identity model, OpenID. We gave an analysis of the most important security questions related to the federated identity model and how OpenID deals with them.

The OpenID standard does not define how to implement the authentication process. This leaves space for risky identity provider implementations of weak authentication, which are vulnerable to many attacks described in this paper. Exploiting such vulnerabilities has serious implications in OpenID, such as identity stealing. Thus, we feel that OpenID should be stricter about the level of security required to implement and not leave it fully to implementers. As a great solution to secure OpenID implementation, we think that the InfoCard standard should be integrated with OpenID. This way, OpenID implementations get the best from both standards, namely strong security, privacy, usability, scalability and openness.

REFERENCES

[1] E. Maler and D. Reed, "Options and Issues in Federated Identity Management", *IEEE Security & Privacy*, vol. 6, no. 2, p. 16-23, 2008.

[2] SAML XML.org, http://saml.xml.org/wiki/sp-initiated-single-sign-on-postartifact-bindings

[3] SAML XML.org, http://saml.xml.org/wiki/idp-initiated-single-sign-on-post-binding

[4] JanRain Blog, http://blog.janrain.com/2009/01/relying-party-stats-as-of-jan-1st-2008.html

[5] JanRain Blog, http://blog.janrain.com/2008/11/openid-user-experience-data.html

[6] Google Code Blog,

http://google-code-updates.blogspot.com/2008/10/google-moves-towards-single-sign-on.html

[7] Windows Live Blog, http:// winliveid.spaces.live.com/Blog/cns!AEE1BB0D86E23AAC!1745.entry

[8] OpenID Blog, http://openid.net/2009/01/28/paypal-joins-openid-foundation-board-as-we-enter-2009/

[9] OpenID Authentication 2.0 specification, http://openid.net/specs/openid-authentication-2_0.html

[10] Hyun-Kyung Oh, Seung-Hun Jin: „The Security Limitations of SSO in OpenID“, *10th International Conference on Advanced Communication Technology*, vol. 3, p. 1608-1611, 2008.

[11] Eastlake 3rd, D., Reagle Jr., J., and D. Solo, “XML-Signature Syntax and Processing”, *World Wide Web Consortium*, 2002.

[12] OpenID Provider Authentication Policy Extension 1.0, http://openid.net/specs/openid-provider-authentication-policy-extension-1_0.html

[13] J. Hodges, “Technical Comparison: OpenID and SAML - Draft 06”, 2008, http://identitymeme.org/doc/draft-hodges-saml-openid-compare.html

[14] MSDN Magazine http://msdn.microsoft.com/en-us/magazine/cc163615.aspx