Using OpenID As An SSO Protocol

Gareth Sime, gsim760@aucklanduni.ac.nz

October 22, 2015

Abstract

Single sign-on is a property of authentication systems that specifies that users should only need to log in with one identity and should only need to enter credentials once. SSO is not a property of standard OpenID. However, with modifications to the way OpenID is implemented, it could be used this way. I discuss whether this is desireable and highlight some of the benefits and limitations of using this modified OpenID protocol.

1 Introduction

Identity management is difficult, both from the perspective of users and developers. In particular, users find it difficult to remember username and password combinations for each service they use (password fatigue). Users also find it tiresome to log in to each site individually every time they start a new session. The average user types about 8 username/password combinations per day and has around 25 different accounts with various services[9].

The goal of single sign-on systems is to aleviate problems associated with password fatigue and to allow users to re-use personal information that they have already provided to other services.

OpenID is an authentication protocol for decentralized identity management. This allows users to establish an identity at their choice of Identity Provider (IP) and reuse this identity across a range of Relying Parties (RPs). By comparing the OpenID specification[1] to our definition of SSO, we can see that OpenID is not SSO.

In Section 2 I define what single sign-on is and what is required of a protocol before it can be said to be single sign-on. Section 3 gives an overview of OpenID and describes it from the users' perspective. In Section 4 I discuss whether or not using OpenID as an SSO protocol is desireable. The concept of an identity enabled browser, as proposed by [12] is examined in Section 5. Section 6 proposes how we might change our use of

OpenID to conform with our definition of SSO and describes some of the benefits and limitations of using OpenID this way. Finally, in Section 8 I conclude my report and give some recommendations.

2 Single Sign-On

Single sign-on is a property that some authentication protocols have. I define single sign-on to mean the following:

- Users are only required to use one identity (they may optionally use more).
- Users should only need to enter credentials once per session per identity.
- Users have appropriate access to all of the services within the given realm.

Any authentication system which fits these criteria is considered to be single sign-on. Alternative definitions of SSO have been proposed but I have chosen this definition as it appears to be the most common, both in the general public[3][4][5] and in the academic community[10][8]. In particular, I will be following the terminology given by [10].

3 OpenID

OpenID is a web based authentication protocol[1]. It allows some services to act as Identity Providers and some to act as Relying Parties. I will largely be using the definitions of Identity Provider and Relying Party given in the OpenID specification.

Users establish an identity at an Identity Provider. An Identity Provider uses some authentication mechanism to allow users to log into their service. Examples of common Identity Providers are Google, Yahoo!, PayPal and IBM.

Relying Parties require users to authenticate to them in some way. To accomplish this, they allow users to sign in using their existing identities on any Identity Provider. This means that users don't need to remember another set of usernames and passwords and often don't have to go through the tedious effort of creating a new identity at every service. Relying Parties rely on Identity Providers to securly authenticate users.

3.1 User Flow

For users, the OpenID protocol generally appears as in the simplified screen diagram shown in Figure 1. Each screen can be seen with their respective URLs above them. In this concrete example, CoolWebsite is the RP and Google is the IP. In the general case,

the user would be redirected to whichever IP corresponds with what they entered as their OpenID URL. The diamond represents a descision point for the IP.

A user visits a Relying Party and, when prompted to log in, enters their OpenID URL (an OP Identifier or User-Supplied Identifier). The RP then redirects the user to their Identity Provider, where they log in with some authentication system chosen by the IP. Users are asked by their IP if they wish to grant the RP certain *claims* (pieces of information about their identity). How a user authenticates to the IP and approves claims is up to the IP in question and is not specified by OpenID.

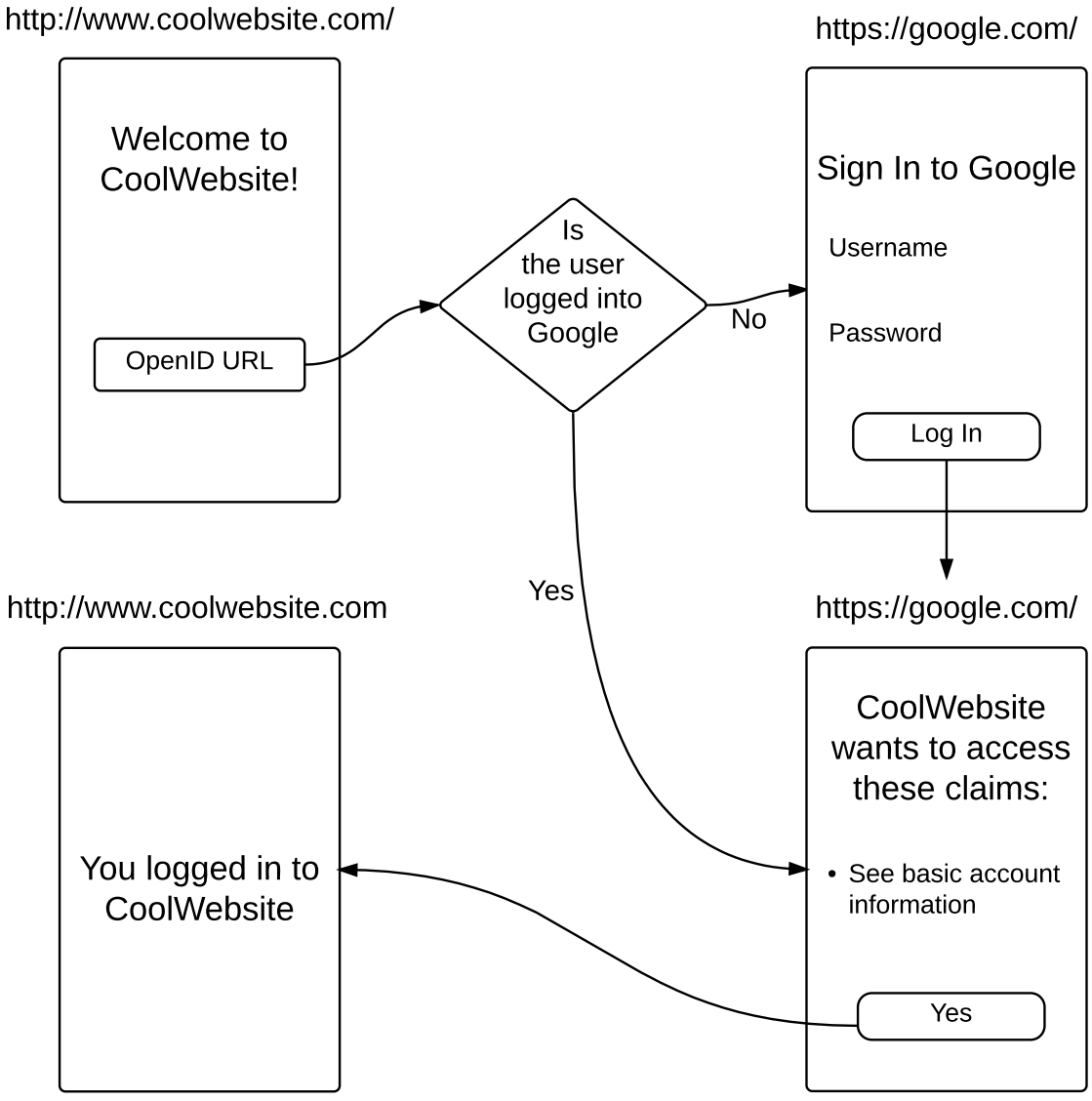

Figure 1: Screen Diagrams depicting the user view of OpenID

3.2 Why OpenID is not SSO

By our definition of SSO, a user is only required to enter credentials once per session. We can see from the screen diagram that before the user can sign in to a Relying Party, they must enter an identifier. This identifier may be specific to that particular identity or it may only specify the Identity Provider they wish to use. This URL can be considered a credential in a similar way that a username is. Users must enter this every time they try to sign into a service. As such, OpenID fails to meet our specification of SSO.

An Identity Provider is not required to remember a user's session. In the real world, IPs often do remember users' sessions, but there is no requirement to. This also breaks the rules of SSO regarding one-time entry of credentials per session. For example, if a user signed in to RP1 using their IP, and then tried to sign in to RP2. There is no requirement that their IP will remember who they are when they try to log in from RP2.

4 Desirability of using OpenID as SSO

The immediate reason we would consider using OpenID as SSO is because it is already widely known and supported. The OpenID Foundation claims that there were over 1 billion OpenID accounts by the end of 2009. However, there are some caveats to the use of OpenID.

[12] attempted to evaluate why users refuse SSO by performing an empirical study of OpenID. The results of that paper could be valuable when considering whether to use OpenID for SSO purposes or not. It must be noted that the authors of that paper used a different definition of SSO. Their definition is that a Web SSO system is any system that separates the Identity Provider from the Relying party. By their definition, OpenID is already SSO by default. As such, we need to be critical in deciding which of their findings, if any, are still relevant to this report, given that their definition of SSO is slightly different to mine.

The authors of [12] point out that people using OpenID have issues with phishing and poor understanding of the protocol. Other points that the paper raises are more relevant to SSO in general, such as users' concerns related to single point of failure problems, release of personal information, and the trustworthiness of RPs. [7] also expresses concerns regarding phishing attacks. Both papers assert that this is one of the primary concerns for OpenID and that any successful SSO system should have some mechanism to allow users to detect phishing with a high level of certainty. [12] proposes that the best method of doing so is to build OpenID support into the browser.

[7] provides a thorough analysis of using OpenID in its current state for enterprise SSO. While their definition of SSO is also slightly different from mine, their analysis remains

useful in pointing out some of the problems with using OpenID for this purpose. They have used a formal modeling approach to analyse how users interact with OpenID and OpenID form a system perspective. They have shown that formal modeling of protocols can be very useful in revealing and studying security risks. The authors have discussed how similar approaches can be taken to analyse other technologies as well.

[7] points out that some form of provider whitelisting is crucial to using OpenID for enterprise. This is because, in an enterprise setting, the RPs need to have strong confidence that the user is indeed the person that they claim to be. If any IP could be used, then an attacker could simply use their own IP and claim to be whomever they like.

5 Identity Enabled Browser

[12] proposed that browsers should support OpenID identity management. Their original Identity enabled Browser (IDeB) prototype was a Wizard of Oz tool used to study how OpenID could be made more usable by enabling browser support for it.

The goals of making OpenID browser supported are as follows. Browser support will encourage users to use OpenID and it will encourage RPs to support OpenID authentication. Managing multiple OpenID identities will be simplified. Phishing scams will be more difficult to achieve as the browser could handle authorization for users (browsers could communicate directly with IPs).

The authors claim that their prototype helps users recognise phishing attacks. They do this by shrinking and fading out the browser window and having a popup over the top, something they assert websites alone cannot accomplish. However, this was never actually verified. Users were only asked to identifiy a phishing scam targeted at the traditional OpenID login and were never asked to identify a phishing scam targeted at their prototype.

The prototype was only a Wizard of Oz demonstration, so it would not make sense to formally analyse the system itself, as was done in [7]. However, we can still model the user interaction using the form storyboarding technique described in [7]. This representation can be used in the future to reason about some security risks that may have been opened up by this browser extension.

One of the benefits of using the proposed IDeB design is that it appears to conform with my definition of SSO. While it is implied by the authors that users must sign in to their IPs each time they start a new session, it is not explicitly stated. This could cause security issues when using shared computers as another user may be able to access the original user's accounts.

6 Modified OpenID

Modifying browser behaviour can be difficult to achieve, particularly with regard to getting everyone to agree on a standard. As such, I am proposing another method to make OpenID conform with my definition of SSO, called Modified OpenID (mOpenID). My intention is to propose a method of using OpenID which fits reasonably well within what is currently in practice.

The most difficult problem to solve with regard to making OpenID single sign-on is the number of times credentials are entered. In order to make OpenID single sign-on, I propose the following changes.

We need a way so that users do not have to repeatedly enter their OpenID identifier. One possible solution to this is to specify that each Relying Party only supports one Identity Provider. This means that users will not need to enter their identifier as the RP will already know where to redirect them for authentication.

Mandating that Identity Providers must remember a user for a whole session is a simple change, particularly since most IPs do this already. It is necessary if we are ever going to use OpenID as an SSO protocol.

6.1 Benefits of using Modified OpenID

Using this mOpenID gives us three main benefits. Firstly, we are more compliant with our definition of single sign-on. Secondly, we gain the usability benefits of SSO. Lastly, mOpenID appears to be very similar to what is currently used in the real world. This means that it would be relatively straight-forward to implement.

mOpenID fits with the idea of IP whitelisting proposed by [7]. In the case of mOpenID, there is one trusted IP (effectively a whitelist of length one). Each RP would pick an IP that they trust to provide accurate information and to properly secure their users' identities.

One of the main recommendations made by [12] is that RPs should make it obvious that users can sign in using their existing OpenID identities. In particular, they found that many people were confused by the presence of OpenID IP buttons (mistaking them for advertising) and many tried to enter their username/password in the OpenID identifier field. mOpenID addresses the problem of users misunderstanding the OpenID identifier field. The RP already knows which IP the user must use so there is no need to ask.

6.2 Limitations of using Modified OpenID

When using mOpenID, users will only ever have one choice of Identity Provider. This creates issues as getting everybody under one definitive provider would be difficult. It also goes against the entire philosophy and intended use of OpenID. The point of the OpenID protocol was to allow for distributed IPs, particularly in environments where RPs have not established a trust relationship with IPs.

mOpenID does not address the issue of phishing attacks. This could be a showstopper for organisations and users who need to be absolutely certain about where they are signing in. After discussion with users on the matter of phishing, [12] found that none of the users would consider using OpenID if they knew that phishing was possible.

Having only one IP also calls into question the matter of multiple identities. Some users may wish to use a variety of identities to authenticate with a range of RPs. In order to support this, the IP would need to be capable of managing multiple identities at once or allow users to sign out of one identity and into another. However, this may cause confusion as users may not remember which identities they have used with each RP (although, this could potentially be stored by the IP as well). This may pose similar issues to those raised by [12] regarding account-linking.

The ramifications of forcing IPs to remember users' sessions are reasonably well understood as most IPs already do this. However, session management in general has its own security concerns associated with it[11][13][2]. The Open Web Application Security Project has identified session management as being the second biggest concern for web applications today[6].

7 Discussion

It appears that there are relatively few benefits of using mOpenID compared to the number of caveats associated with it. While some of these changes would be simple to make, they generally go against what OpenID was designed for.

For most users, the biggest change would be not having to enter their OpenID identifier. However, entering in an OpenID identifier at each RP may not be a difficult task once users have used OpenID for some time, so users do not stand to gain much in usability from this change but they do loose a lot in their choice of IPs.

The changes proposed by mOpenID are almost entirely because of the strict requirement that credentials are only entered once. This stringent specification does not apply well to OpenID and it seems to decrease the usefulness of the protocol rather than increase it. OpenID may not be strictly SSO, but it is in many ways still a very useful protocol that acomplishes similar goals.

8 Conclusion

I have discussed some of the reasons that OpenID does not comply with my definition of single sign-on. Changes to the way OpenID is used have been proposed and the benefits and limitations of doing so considered.

I have discussed IDeB and I recommend that it needs further investigation and modeling before it could be used.

Future research on mOpenID could build the proposed system and then empirically evaluate its effectiveness. Techniques discussed by [7] could be used to analyse mOpenID using formal models.

Overall, it is my opinion that the limitations of using mOpenID outweigh the benefits. As such, I do not recommend that OpenID be used in this way and I do not believe that further research on the matter is required.

References

[1] OpenID authentication 2.0. `http://openid.net/specs/openid-authentication-2_0.html`. (Visited on 10/11/2015).

[2] Session management cheat sheet - OWASP. `https://www.owasp.org/index.php/Session_Management_Cheat_Sheet`. (Visited on 10/12/2015).

[3] Single sign-on - Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Single_sign-on`. (Visited on 10/11/2015).

[4] What is single sign-on (sso)? - definition from techopedia. `https://www.techopedia.com/definition/4106/single-sign-on-sso`. (Visited on 10/11/2015).

[5] What is single sign-on (sso)? - definition from whatis.com. `http://searchsecurity.techtarget.com/definition/single-sign-on`. (Visited on 10/11/2015).

[6] Owasp top 10 - 2013. `http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf`, 2013. (Visited on 10/12/2015).

[7] Bellamy-McIntyre, J., Luterroth, C., and Weber, G. OpenID and the enterprise: a model-based analysis of single sign-on authentication. In *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International* (2011), IEEE, pp. 129–138.

[8] De Clercq, J. Single sign-on architectures. In *Infrastructure Security*, G. Davida, Y. Frankel, and O. Rees, Eds., vol. 2437 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2002, pp. 40–58.

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

[10] Pashalidis, A., and Mitchell, C. A taxonomy of single sign-on systems. In *Information Security and Privacy*, R. Safavi-Naini and J. Seberry, Eds., vol. 2727 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 249–264.

[11] Subashini, S., and Kavitha, V. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications 34*, 1 (2011), 1–11.

[12] Sun, S.-T., Pospisil, E., Muslukhov, I., Dindar, N., Hawkey, K., and Beznosov, K. What makes users refuse web single sign-on?: an empirical investigation of OpenID. In *Proceedings of the Seventh Symposium on Usable Privacy and Security* (2011), ACM, p. 4.

[13] Zissis, D., and Lekkas, D. Addressing cloud computing security issues. *Future Generation computer systems 28*, 3 (2012), 583–592.