Consumer Third Party Authentication: Challenges and Potential Solutions

Craig H. Wittenberg, Microsoft Corporation
craigwi@microsoft.com

April 2011

Abstract — Usernames and passwords are an approach to authentication fraught with well documented problems. Third-party authentication (e.g., via OpenID) is meant to address many of those problems but introduces a different set challenges. This brief paper describes some aspects of the problem space, enumerates three key challenges and offers three possible solutions – all with the desire of prompting a lively dialogue.

I. Background

For this discussion *consumer authentication* is the means and precision by which a user reestablishes a session with a device, application, web service or site. We informally refer to this as "login" and for the purposes of this document the terminology used will be "login" to a "site". Additionally, login involves only the presentation of an identifier and some proof that the user is already known to the site by that identifier.[1]

Most of us login into many sites with a username and a password. The username is a type of identifier. There are many kinds of identifiers, including email addresses, account numbers, random numbers, and well-known numbers. Different types of identifiers have different properties, for example the degree to which they enable identification of the user herself. Such properties relate to challenges discussed below.

The purpose of the authentication process is to establish that a return user is the same person or entity as the one previously logging (e.g., with the same username/password). The *strength* of authentication is orthogonal to the discussion – here any solution will naturally need to take into account such requirements.

Two-party authentication involves just the user and the site. This approach is widely deployed and has many, well documented problems [1], including the difficulty that site owners face in managing user accounts. However, several aspects of the two-party approach are desirable, not the least of which is the ability to keep different contexts separate (or, put another way, the avoidance of one large "honey pot" where all of a user's data resides).

Third-party authentication involves proving to a third service (referred to as authentication provider) the binding of the identifier to the user, followed by transmission of the results of that of the process to the site. The site then accepts or rejects this attempt to login. This type of authentication is also known as "federated" authentication.

Technically there are many ways to achieve consumer third-party authentication. OpenID is one well-known approach. Windows Live uses a proprietary but functionally similar approach. All (or nearly all) approaches depend on browser redirection; the vagaries and relative lack of security of browser redirection have been documented (e.g., phishing; [2]) and will not be discussed here. There are also many user experience (UX) problems which have received attention recently.

II. Challenges

This section focuses on three critical, but frequently overlooked challenges in three-party solutions. Solutions to the overall problem of consumer authentication need to address the challenges below as well as those documented elsewhere:

A. Privacy

While there are many facets to privacy, discussion here concentrates on two major concerns: tracking authentication events and tracking users across sites.[2]

Tracking authentication events: In the case of third-party authentication, the authentication service is contacted each time an authentication occurs. Thus each and every attempt to login can be tracked. This is further compounded in the (expected) case that a consumer uses one authentication provider to login to

[1] Mutual authentication is of course important, and the techniques described in this paper can be applied with the roles of the user and the site reversed.

[2] The association of the user and identifier at a specific site is part of the purpose of login and thus out of scope of this discussion.

many sites. In other words, the authentication provider (sometimes called an identity provider) has broad knowledge of the user's activities based on his or her many authentication events.

Tracking users across sites: The second problem is that of tracking the same user across sites. If a user employs the same identifier at two different sites, then the two sites can trivially determine that the two accounts represent the same user. While there are user benefits that can come from having fewer identifiers – namely, simplicity – the issue really is one of convenience given how hard it is for the average user to create and manage multiple identifiers. Some third-party authentication solutions address this problem (e.g., certain profiles of OpenID), but they typically still allow the authentication provider to track the authentication events.

B. Availability / Customer Support

Logging into a site using a third-party system necessarily means interacting with two systems, not just one[3]. When these systems are on the Internet, there are many things that can prevent the authentication service and/or the destination site from being available or can cause them to respond slowly. In other words, the likelihood that two systems will both be up and running at the same time will always be less than for one system.

Also, when there are problems, whom does the user contact to resolve the issue? The user naturally would start with the site being accessed; the costs to resolve the problem, for both the user and the companies involved, could be quite high. The site that had wanted to "outsource" authentication may end up with unhappy users (cf. [3]).

C. Branding

Given the choice, a bank, online store or other major web site wants to attract users to their site *and* develop ongoing relationships with them – brand recognition. Third-party authentication as done today introduces a second brand – that of the authentication service – into the user's awareness, disrupting the experience with the site and its brand.

Although it seems possible to develop a customized user experience, such customization would have to be done for each and every authentication service in use by a site's users. Such customization would also be expensive for the site.

III. Potential solutions

Below are three solutions that address the challenges laid out above (and some of those documented elsewhere). Each solution takes a different approach and each has different specific qualities.

A. Trust Framework

In the purest sense, a trust framework isn't a technical solution to these problems and thus doesn't address all of the challenges. I mention this approach because there are a number of groups thinking creatively on the topic (e.g., OITF [4], Kantara [5], Open Identity Exchange [6], and US TFPAP [7]), because trust frameworks can help address some of the challenges (privacy and customer support), and because ultimately any solution will likely have some legal or policy elements to it.

To take one concrete case, the possibility that users could be tracked by an authentication provider isn't "solved" by a trust framework, but a framework could be established to impose contractual obligations to protect the data collected and help prevent its misuse (e.g., similar to the way PCI DSS rules work).

B. Shared, Symmetric, Strong Secret

Shared, symmetric, cryptographically strong secrets are commonplace. For example, TLS uses a number of mechanisms to bootstrap to a shared, symmetric, strong secret. Microsoft's Active Directory establishes a shared, symmetric, strong secret as part of the user and machine authentication processes.

Passwords are shared, symmetric, secrets but are generally very weak. Recent studies have shown just how bad they are [8]. In other words, humans are inefficient at generating or managing such secrets.

Consumer passwords managers are not widely deployed but are an attempt to raise the quality of shared secrets and simplify their management. Many passwords managers simply avoid the challenges above because they are actually or effectively a two-party solution to authentication.

The information technology industry has a wealth of experience with shared, strong secrets in other parts of our systems; we should apply these solutions to the area of two-party user authentication. To be able to deploy such solutions seamlessly will also require other changes (e.g., deterministically marking password fields on forms).

[3] Or three instead of two depending on what you count.

C. *Privacy Friendly Login Certificates*

Lastly, there are applications of asymmetric cryptography that could be used to solve the challenges above. X.509 certs can address some of the challenges, but the issuer knows the signature of the certificate (and thus can trivially track the user); challenges in PKI management infrastructure have also not been solved.

Blind signatures, introduced by David Chaum [9] in the early 1980s, offer a way to issue certificates for which the final signature is not seen by the issuer. An example of the application of blind signatures is PseudoID [10].

There are two related technologies, from Microsoft and IBM, both of which can be applied to these problems and both of which solve other problems as well. The U-Prove technology, available free of charge from Microsoft, utilizes blind signatures [11]. IBM's Idemix [12] is an application of group signatures (introduced by David Chaum and others in the 1990s).

Although in varying degrees, all solutions utilizing these techniques will be three party solutions and address all of the challenges above. Blind signatures avoid the trivial tracking by issuers. Once issued the certificates become strong replacements for usernames and passwords, operating in a two-party fashion that avoids the availability challenge. The client software necessary to manage the certificates can also address the customer support and branding challenges.

IV. Conclusion

In the end, there is a role for all three solutions in consumer authentication, all based on open standards, packaged in easy to use devices, software, and cloud services. Our challenge as a community is to work towards a more trustworthy and secure experience, one feature and component at a time.

The opinions expressed are my personal thoughts and do not necessarily represent the opinion of Microsoft or any other vendor mentioned here. I look forward to our discussion.

References

[1] Bruce Schneier, Real World Passwords, December 2006; http://www.schneier.com/blog/archives/2006/12/realworld_passw.html

[2] OpenID Security Review, October 2009; https://sites.google.com/site/openidreview/issues

[3] Rob Conery, Open ID Is A Nightmare, April 2011; http://blog.wekeroad.com/thoughts/open-id-is-a-party-that-happened

[4] Mary Rundle, ed., The Open Identity Trust Framework Model, March 2010; http://www.microsoft.com/mscorp/twc/endtoendtrust/vision/oitf.aspx

[5] Kantara Federation Interoperability Working Group; http://kantarainitiative.org/wordpress/tag/trust-framework/

[6] Open Identity Exchange; http://openidentityexchange.org/

[7] http://www.idmanagement.gov/documents/TrustFrameworkProviderAdoptionProcess.pdf

[8] http://www.darkreading.com/blog/227700652/phpbb-password-analysis.html

[9] David Chaum. Blind signature system. In D. Chaum, editor, Advances in Cryptology–CRYPTO '83, page 153, New York, 1984. Plenum Press.

[10] Arkajit Dey and Stephen Weis, PseudoID: Enhancing Privacy for Federated Login; http://www.pseudoid.net/

[11] http://www.microsoft.com/u-prove

[12] http://www.zurich.ibm.com/security/idemix/