A WebID Implementation in Pure JavaScript and Flash

Manu Sporny, David Longley, David I. Lehn, and Mike Johnson
W3C Web Payments Community Group
Digital Bazaar, Inc.
Blacksburg, VA 24060, USA
{msporny, dlongley, dlehn, mjohnson}@digitalbazaar.com

Abstract

WebID is a key enabling technology for the distributed social web. It enables cryptographically secure password-less login and personal data storage. Often it takes browser manufacturers a long time to adopt new technologies. Instead of waiting for browsers to provide native programming support and custom user interfaces for WebID, our method implements WebID in pure Javascript and Flash and works on all major platforms today. This enables immediate deployment and rapid adoption of the technology, but still allows browser manufacturers to eventually enhance the experience by providing programming hooks and interfaces.

1. Introduction

If there is one thing that is universal to all websites, it is the login process. Almost every website requires you to create an account, enter your e-mail address, verify your account, and log in before you can use any of the advanced features of the website.

It would be much better if there were a universal login mechanism for the web. Instead of manually creating an account on every website, you would only have to click a button and your browser would automatically log you in and provide your account details. You would never need to remember different usernames or passwords for every site. Your personal data would be stored in only one place, completely under your control, and communicated only to the websites that you allow.

The good news is that there are some very smart people working on these problems. The solution is forming around a specification called WebID[1]. The bad news is that there have remained several problems with the WebID specification that will take the browser vendors years to implement. In this paper, we explain how we have created a way to make WebID work in all the current browsers in use today, including Internet Explorer, using pure JavaScript and Flash.

2. The Website Login Problem

Before we get too involved with WebID, we must identify the problems with website login:

- Most websites require a username and password to use non-basic features.
- It is best to use different passwords on different websites and change the password every couple of months, but hardly anyone follows this advice in practice. It is too difficult to remember all of the different username and password combinations. We tend to use a variety of usernames and a single password, that we never change, on all of our websites.
- Your e-mail address is required on almost every website. Your address information is often required for credit card transactions. Filling out identification forms over and over on the web is time consuming and unnecessary. Computers were supposed to be doing these monotonous tasks for us by now.
- Changing your avatar image or e-mail address requires you to update this information on every website that you use. You should only have to change it once and then that change should propagate to all of your websites.
- If you lose your password or forget your username then you're locked out and usually have to reset your password. That is, as long as you can remember what e-mail address you used to sign up to the site in the first place. At times, you lose access to a particular e-mail address if you have changed e-mail providers.

This problem is not new and many people have attempted to solve it over the past decade. Microsoft tried with Passport and Windows Live ID[2], but failed because Microsoft did not release the technology under an open license. Facebook is doing well with Facebook Connect[3], but requires that you have a Facebook account. Several large websites have adopted OpenID[4], which is better and compatible with WebID, but is still too restrictive for many of the scenarios that WebID addresses. The issue with all of these services is that they do not work with the fundamental architecture of the web, which is distributed. WebID is distributed and is designed to work in concert with the principles that have made the Web flourish – data portability, strong cryptography, data ownership, and decentralization.

WebID solves all of the above problems by providing a single universal identification mechanism that is controlled by you at all times. Just like you have a set of keys to get into your home, WebID provides you with a set of keys to access your favorite websites. Repetitive website registration and multi-site passwords are no more. Changes to your profile can be done in one place and the changes propagate to all your WebID-enabled websites. With WebID, you no longer have to fill out your identification information on multiple websites. You also get the peace of mind that if someone steals your laptop or your WebID keys, you can always just deactivate the keys that were stolen via your WebID provider and continue to use your WebID with a new set of keys. In the real world, if you lose your keys you do not have to change your house, you just have to replace the locks and get a new set of keys. The same analogy applies to WebID.

WebID addresses these problems by providing you with a single web page with a set of computer-generated keys that you can publish anywhere on the Web.

When you go to a WebID-enabled website, it will ask your browser for your WebID and then go to your WebID page to get all the information that is necessary to log in. Your browser, in cooperation with your WebID page, works with WebID-enabled websites to ensure that you and only you can access your account on the WebID-enabled website. The browser does this by using something called Transaction Layer Security (TLS) and client-side certificates. That is, your browser has a very special key that can be used to log into WebID-enabled websites. Nobody else has that key and that key identifies you to the WebID-enabled website.

3. The Problem with Browser-based WebID

Many of the technical hurdles have already been overcome with WebID, but there remained one final piece of the puzzle. We could not move forward with WebID without the agreement from every single one of the browser vendors. Getting browser vendors to agree on new features is a very long, multi-year process. Most of the successful Web technologies get traction outside of the web browser and eventually make it into the web browser in time.

There are two vital browser-based technologies required for WebID to work; the <keygen> element and client-side certificate support. This support has been in Firefox, Safari, and Chrome for quite a while now while Internet Explorer provides its own proprietary ActiveX mechanism. It could be half a decade before <keygen>, and thus proper WebID support, is integrated into IE. This is holding WebID adoption back in a very negative way.

The problem was that TLS implementations in the browser do not surface the proper application programming interfaces to developers that want to modify the connection process – a requirement for WebID to work properly. Asking the browser vendors to allow this would kick off years of work for all parties involved. One way to direct the TLS connection would be to write a custom plugin for the browser, but then you would require everyone that wanted to use WebID to download a custom browser plugin. Developing plugins for all the major browsers would also be a difficult undertaking.

For WebID to be successful, it must work in most of the browsers that are out there already. Unfortunately, the only technologies that are broadly available to implement TLS and client-side certificates are Javascript and Flash.

4. TLS in Pure JavaScript and Flash

Digital Bazaar has implemented TLS in pure Javascript and Flash and released the implementation under a dual GPL/BSD license – the project is called Forge[5]. Forge implements TLS certificate generation and client-side certificates for WebID in pure Javascript and Flash. The primary benefit being that the implementation works in Internet Explorer and every major browser in existence today. The benefits as a result of this pure Javascript and Flash implementation of WebID include:

- WebID now works in a standard way in recent versions of all major browsers, including Internet Explorer.
- WebID now has a consistent interface for selecting WebIDs across all WebID-enabled websites.
- The source code has been released to the Web community as open source software and can be easily integrated by sites like Facebook, Twitter, and others that want to support WebID-based login.

5. The Pure JavaScript WebID Demo

Most of these concepts are best demonstrated via the WebID demo. In the demonstration, you will create a WebID and then use that WebID to log into a website. Please note that this is all very preliminary and will not work in some older browsers, like IE6 and IE7, or browsers with very slow Javascript engines. For best results, use a recent version of Google Chrome. That said, it should work in Firefox 3.5+, Chrome 5.0+, Safari 4+, and IE8+.

You will also see security errors when you click on the two links below. This is because we have not purchased SSL Certificates for either site. If we had purchased SSL Certificates for either site, you would not see the warning. This is a demo after all, so just ignore this minor nuisance for now. When WebID is deployed in a production setting, the sites would have real SSL Certificates.

First, you will need to get a WebID. We have setup a demonstration WebID provider[6]. Go there, ignore the warnings and accept the certificate, enter your name in the “Name” field and click Create. It will take several seconds to generate the public and private key pair for your certificate. Public/Private key certificates are at the core of WebID, and thus require quite a bit of computation to generate. The certificate generation will be faster on browsers that have very fast Javascript engines. Chrome will only take a few seconds whereas Firefox may take 15 seconds or more to generate a certificate. You may get a warning that the script is taking a long time to complete, do not stop the script, it is doing a very hard math problem and will eventually finish. Once the certificate is generated, you will see an entry pop up in the “Your WebIDs” part of the interface with your name attached to it.

After you have generated your certificate, go to the WebID-enabled website[7] and ignore the warnings and accept the website certificate. To log into the website, click on “Digital Bazaar WebID” and then click on the blue “Select” button beside the WebID that you would like to use to identify yourself to the website. If everything goes well, you should see a Welcome screen.

This small implementation[8] demonstrates how easy it is to generate a WebID and log into a WebID-enabled website. The demo proves the following WebID concepts:

- Generating a X509 Security Certificate using pure Javascript.

- Storing the certificate using Flash local storage, so that all of your browsers have access to the certificate.
- Implementation of a WebID provider using pure Javascript and HTML.
- Secure retrieval of WebIDs on any website using an iframe.
- Selecting one or more Certificates from Flash local storage.
- Initiating a TLS connection to verify your WebID using pure Javascript and Flash.
- Successful login using WebID via TLS.

There are a few more features that we need to implement to make the demo more useful, namely:

- Adding more features to the WebID management page like certificate revocation, OpenID support, and adding things like friends, photos, and address information.
- Usability improvements in browsers like Internet Explorer and Firefox.
- Expressing the WebID using structured data (like RDFa or Microdata).

There are also a few remaining problems with this approach:

- One-time use WebIDs are necessary for login via public terminals. A WebFinger-like discovery mechanism coupled with an OAuth-style key exchange could be used to do public terminal logins.
- If there are going to be multiple WebID providers, the NASCAR problem exists and would need to be addressed via XAuth[9], a WebFinger-like discovery mechanism or native browser implementations.
- Browsers without Flash support, such as many mobile devices, will need to use alternate communication methods such as WebSockets.

Browser-based implementations could eventually provide:

- A better user experience like Firefox Weave for selecting different identities when logging into a site.
- An API so that websites could choose to use the pure JavaScript implementation or use the browser API to perform login and discovery of preferences.
- A more integrated experience between the WebID provider when creating things like one-time WebIDs at public terminals.

6. Conclusion

It is possible to implement WebID in pure JavaScript and Flash. Waiting on the Web Browser manufacturers is a flawed strategy, unless they all buy into the technology in the very near future. Giving the browser manufacturers time to see if WebID will succeed in the market will reduce risk for them while helping most of the interface issues to be worked out over the next few years. Eventually, native in-browser implementations for WebID will provide an even better experience than the pure JavaScript and Flash approach.

Bibliography

[1] Stéphane Corlosquet, Manu Sporny; et al. The WebID Specification February 10th 2011. W3C Editors Draft.
URL: http://www.w3.org/2005/Incubator/webid/spec/

[2] Wikipedia. Microsoft Passport as of April 2011.
URL: http://en.wikipedia.org/wiki/Microsoft_Passport

[3] CrunchBase. Facebook Connect Description as of April 2011.
URL: http://www.crunchbase.com/product/facebook-connect

[4] Recordon, Fitzpatrick, Hardt, Hoyt; et al.
The OpenID Authentication 2.0 Specification December 5, 2007 .
URL: http://openid.net/specs/openid-authentication-2_0.html

[5] Digital Bazaar. Forge Source Code.
URL: https://github.com/digitalbazaar/forge#readme

[6] Digital Bazaar. WebID Provider Demo Website.
URL: https://webid.digitalbazaar.com/manage/

[7] Digital Bazaar. WebID Client Demo Website.
URL: https://payswarm.com/webid-demo/

[8] Digital Bazaar. WebID Demo Source Code
URL: https://github.com/digitalbazaar/webid-demo#readme

[9] XAuth
URL: http://xauth.org/