The Case for Cloud Identity Selector Services

Avoco Secure Ltd
Dr. Steve M. Hitchen

info@avocosecure.com

US: +1 415 839 9433

International: +44 207 851 6070

www.avocosecure.com

Using digital identities (*e.g.* Information Cards, OpenIDs, *etc*.) to identify users to resources, such as websites and services, is greatly facilitated through the concept of an *identity selector*. The identity selector displays to the user their digital identities in the form of some familiar metaphor, such as a pictorial representation of a card, and permits the user to choose which identity is to represent them through simply clicking on the appropriate image. Behind the scenes, the selector orchestrates communication between the resource (known as the *Relying Party*), the user and the *identity provider*, the latter being the system that serves up the required identity information, such as the user's name, email address, and other identity-based claims. Figure 1 shows diagrammatically the interactions between the principal components.

Historically, selectors are components that are installed on the user's desktop or device. The *de facto* standard for these client selectors is Microsoft Windows CardSpace, but, at the time of writing (March 2010) others, based on open source code, are available for Windows and other desktop operating systems and also the Apple iPhone. Recently however, a totally Cloud-based selector, which requires no desktop or device installation, has become available from Avoco Secure. This article compares the merits and features of both approaches to identity selectors. The four main areas to be examined are:

1. Identity Usability
2. Compatibility and Maintenance
3. Features
4. Security

Identity Usability

Here, we are considering the factors that make it easy to use digital identities, from the viewpoint of both the end-user and of the Relying Party. These factors are extremely important: ultimately, they determine the scale of take-up of the use of digital identities.

Leaving aside specific user interface issues, such as the use of the visual card metaphor, which apply equally to both client-side and Cloud-based selectors, from the user's perspective a significant usability issue is availability of their digital identities. The trend for some time has been that a given user will want to access on-line resources using a variety of different desktops and/or devices. For example, a user may use a desktop machine at work, a laptop at home and a smart phone or similar device while travelling. If their digital identities can be made available regardless of the underlying system, then users are far more likely to make use of them.

By its nature, a Cloud-based selector has this highly desirable feature built-in: each user's identities are stored in the Cloud and are available from any device or desktop, regardless of the operating system or browser.

In contrast, to ensure that their identities are available with client-based selectors, a user would have to ensure that a selector is installed on every desktop or device they use. Even if

a suitable selector is actually available for the device(s) or operating system(s) in question, the user must first find it. Furthermore, installation of client software relies on both the assent and technical competency of the end user, either of which may well be absent.

Once the user has completed the task of installing a selector on all required devices, their identities would then need to be exported as needed and then imported onto each device; this latter step would need to be replicated each time the user obtains a new or modified identity. Clearly, these steps may be neither desirable nor (often) feasible if the client machine were a public or company one.

Overall, these requirements form a significant impediment to mass uptake of digital identities.

From the viewpoint of the relying party, most desktop selectors can be accessed easily by invocation of the installed selector through a standard *application/x-informationCard* object. However, because relying parties cannot rely on the presence of a desktop selector they will generally only offer the use of signing-in with a digital identity as an option, if at all. On the other hand, if the relying parties either hosted a Cloud selector or made use of a Cloud selector service, it could ensure that users could consistently sign in with a strong digital identity.

Compatibility and Maintenance

With several different client side selectors being available, each potentially with different features, the issue of possible conflicts arises. For example, one relying party may decide to require features provided only by a specific selector - an example of this could be the Action Cards supported by the Azigo client selector. The user then goes ahead and installs the required selector to access the site's features. However, other relying parties may require features that are not provided by the selector that has now been installed. This leads to a conflict as, currently, only one selector may be installed and active at a time on a given machine or device.

The use of a Cloud selector completely avoids this problem, as the relying party can specify which Cloud selector to use, and offer the choice of multiple selectors, as required.

Because desktop selectors interact directly with both the browser, usually as a browser plug-in, and with the client operating system, issues of compatibility often arise. It is always difficult for plug-ins to be written so that seamless support for multiple browsers can be achieved, particularly as the browser may change through updates, without notice, causing the plug-in to fail unexpectedly. With a Cloud selector, because the underlying selector engine is separate from the user interface code, and the latter is in the form of web pages, it is a simple task to ensure compatibility with all browsers.

Of course, the problems of client software maintenance and upgrades are other issues that are also avoided with a Cloud solution.

One further aspect of the use of Cloud selectors in this area should be mentioned: the single point of access of a Cloud selector permits fast, universal updates, ideal for adoption of new

developments. This inherent capability for rapid uptake of new methods actually feeds the development process, by encouraging advances in the fields of digital identity and on-line verification that can then be rapidly deployed. In contrast, the inevitable problems of upgrading of client-based selectors tend to inhibit adoption of new developments.

Features

A Cloud selector can more easily provide enhanced features than a client-based solution, without causing the compatibility issues noted above. Because a Cloud selector is communicated from the relying party through a standard HTML link with query string parameters, it is simple to implement by a relying party and this also makes it straightforward for the relying party to communicate extra information to the selector, over and above that possible through the standard plug-in interface. For example, it is possible for the relying party to specify that only cards that are accessed using certain types of user authentication are acceptable; this could be used, for example, to provide enhanced security, by specifying that only cards that require authentication with an x509 certificate can be used.

Similarly, when a card permits the use of more than one authentication, such as x509 or password, a Cloud selector can permit the more intuitive use of a password, when the certificate is unavailable, rather than only allowing use of a password when the STS endpoint that handles x509 is unavailable, as is the current client selector standard.

In essence, the expansion capability of Cloud selectors is unlimited.

Security

Superficially, it may seem that that use of a client selector is more secure that a Cloud one, for the reason that the client selector is installed on the user's desktop, and does not rely on authentication to a Cloud system, with the latter being potentially susceptible to phishing attacks, server hacks, *etc*.

However, from the viewpoint of the relying party, the question of security of the selector is not so straightforward; with a client selector, the relying party has no control over what selector is used. This means that the relying party has no option of specifying, for example, a selector that meets particular security requirements.

Furthermore, a common argument that installation of the selector is in the user's hands, and is therefore secure, has no meaning: it is the relying party that is making the security demands, not the end-user, and until the end-user has been verified, the relying party cannot trust them, or any selector that has been installed by them.

With a Cloud selector, the relying party can specify exactly which Cloud selector to use; furthermore such a selector can be verified separately by the relying party, including use of out of band techniques. In this way, a Cloud selector can potentially better satisfy the security requirements of a relying party then a client selector.

Of course, not all relying parties are concerned with the security of the selector from this viewpoint. And, from the user's perspective, there will be the concern of storing their digital identities on-line. In fact however, through incorporating good security practices, including anti-phishing methods and mandatory SSL, a Cloud selector easily be made to be at least as secure as the websites that will use it, so the selector is not some sort of weak link. Also, users may also use strong authentication methods, such as, voice biometric, out-of-wallet systems and x509 digital certificates, to access their Cloud accounts, where the highest security is required.

Finally, is should be noted that many of the currently-available selectors, based on open source code, do, in fact, store the digital identities on-line, making them not quite what they seem.

Conclusions

Cloud selectors offer so many positive feature and benefits, both to relying parties and end users, that their uptake seems inevitable if the use of digital identities for on-line authentication is to see mass consumer uptake: requiring users to install client software in order to access web resources seems both illogical and anachronistic.

One alternative to Cloud selectors that may see use in some vertical markets is selectors that are either located in hardware tokens, or that can make use of identities stored in such devices. These can potentially address some of the pitfalls of client-based selectors (*e.g.* identities are available on a wider range of machines), but they still fall short of the universal nature of the Cloud approach.