Single Sign-On: Myth or Reality

Mark Wilcox – mark@mjwilcox.com

Introduction

The concept of single sign-on is one of the greatest quests in network computing. Some might even argue that it is the Holy Grail of network computing since it has been a concept that has been discussed for years, but has yet to find a common solution.

In this presentation we will discuss what SSO is, the risks and rewards of using such a system and the current progress towards solving this problem.

What is Single Sign-On

Single Sign-On is the process by which an user authenticates to a central authentication service and after that they do not have to provide their authentication credentials again. SSO is a key cornerstone of managing online identity.

The key concept of this process is "not providing authentication credentials again". By authentication credentials we normally mean userid and password, though it could also mean other forms of proving your identity such as a digital certificate or your thumbprint.

Within the past decade we've made great strides at providing a framework for doing this type of work, most notably Kerberos and the Lightweight Directory Access Protocol (LDAP).

However, Kerberos, which is a true SSO system requires all systems that wish to interoperate in a Kerberos environment be modified to use Kerberos for reasons we'll go over in a later section. Because of this requirement you do not see many "Kerberized" applications on the market today. In particular your average Web browser does not support it and there's no support for it in HTTP so, there's no way for standard Web services (both traditional things like Web pages and those mythical XML-RPC services).

LDAP, which is the standard for directory services, provides a great back-end database to power standardized SSO, but by itself it does not do SSO. With LDAP all you really get is a standard way to share passwords and attributes about a person, there is no way with standard LDAP for one application to know that you've already authenticated previously.

So the real glitch in SSO is how do you let applications know that you've already authenticated.

It's not just who you are, but what can you do

Most people have a tendency to focus entirely on only on the authentication portion of the identity problem. What they fail to take into account is the descriptive elements that we need to make valid authorization statements.

For example, just because I know your name, doesn't really help me in determining in making a judgment about whether you can access a particular object or perform a particular task, except in those cases where we can make blanket assumptions (we allow all authenticated users into a system).

And too often we focus so much effort on the authentication question that we totally ignore the fact that in many cases the security issue we're trying to solve is an authorization problem, not an authentication problem.

For example we have spent a lot of money and time into making accurate authentication determination on airplanes since September 11, 2001. However, the attacks on that day were not really authentication failures, they were authorization failures. We knew it was "Mohammed Atta" on that plane. Perhaps we didn't know he was a terrorist and we have some type of belief that at some point if we do enough identity checks we'll catch the terrorists at the gate (as if someone believes a terrorist will be carrying "Hi my name is Abe. I'm with Al Queda" card), when in fact what we should be doing is just saying "unless you're a member of the flight crew you can't get into the cockpit".

And putting the focus on authenticating the flight crew, while doing minimal 'authentication' checks on general passenger population when they board. Basically there's no reason for you and I to have to prove whom we are to fly, except for one reason. That is to prevent us from reselling our tickets.

The point of this digression is this – while authentication is important, it is not nearly as important as authorization. And you should be working on authorization solutions that only require the minimal amount of authentication information to perform their function because it improves overall customer service and privacy.

Do I want SSO

If you can combine an SSO solution with a standard authorization protocol to form a complete electronic identity management solution then there a number of key benefits.

The primary benefit is that it improves the overall experience of the customer (they only have to remember one username/password combination) and reduces help-desk support calls because an overwhelming number of support calls are about people forgetting passwords.

The secondary benefit will be that users could potentially have a greater control over their data and who it is shared with. Of course this has as many socio-political barriers as it does technological ones.

For example, most people probably would rather not have their personal information shared with anyone without their consent; yet, this would pretty much destroy the marketing industry as we know it (and if you think the RIAA is testy when it comes to digital music, wait until you see the marketing industry up in arms).

The third benefit is that it allows you as a network administrator to set up policies that can better manage the users and access controls in a central location instead of having it scattered about among several systems.

The Pitfalls of SSO

The number one pitfall of SSO is that if you have a single password, then I only have to guess that password to get access to any system. There are ways to make this harder to do. One way is to use something like digital certificates or biometrics (things like fingerprint readers and retinal scanners) which require more skill and effort to forge. Unfortunately the infrastructure required to pull this off is so vast that it will be years before this is practical. Instead you should randomly generate long passwords and have users write them down.

Yes you read that correctly. Write the long (as in 15 characters or more) password down on a slip of paper and tell them to treat it like their credit card number or cash.

By making it long and random you make it virtually impossible to guess and an implausibly long time to brute force crack.

If users right it down and can treat it like a credit card number, then there is much lower risk of having it stolen.

The second pitfall of SSO is that if the authentication service goes down, you lose access to everything. While it is certainly true you could provide local access for certain accounts (say the super-user account), this is a back-door that has its own pros/cons that have to be weighed before opening.

The third pitfall is that by centralizing this management you do make it easier to track what people are doing on their computers. This is potentially dangerous if improperly utilized by anything from mass-marketers to corporate management to the government. One solution to this is to use the concept of translucent databases to authorize transactions. Essentially what you do is that instead of storing an username and IP address for transactions, you only store an MD5 hash (that is combined with some secret) that makes it harder for someone who doesn't know how that hash was created to abuse the system.

Federating Authentication/Identification

In the past year a term called "Federated Authentication" has emerged in the SSO world. What it means is that every organization already has a way of identifying users for their

use and more importantly, it will be impossible to expect everyone to change how they wish to authenticate users.

In part this is impossible because it would require changing a vast amount of technical infrastructure, a cost that nobody is willing to swallow in these uncertain economic times.

And perhaps more importantly, nobody wants to give central authentication control to anyone, which Microsoft found out the hard way when they initially tried to push Passport.

Federating Authentication == Real World

At first federated authentication sounds complicated – expecting organizations to trust the authentication of other systems to complete transactions.

But this is something that we do all of the time in the physical world.

Everyone contains several different forms of authentication. You have a driver's license, credit cards, insurance cards, a Sam's or Cosco card and perhaps even a student id card.

And depending upon the context in which you're operating, you delegate the authentication to a trusted 3rd party.

Who the trusted 3rd party is depends upon how much trust is required for the operation to succeed.

For example if I wish to purchase student tickets at a movie theater, my student id will work. But if I wish to enter a foreign country they will require at least my passport.

And in general nobody standardizes how each of these entities verifies my actual identity, but also everyone understands that it requires more effort to obtain a passport than it is a student id card.

The Quick Guide to Kerberos

Kerberos is the standard for building SSO outside of the Web world. In fact even Microsoft has adopted (or embraced & extended if you prefer) Kerberos as the default authentication system for Windows 2000 based networks.

And while Kerberos itself has not made it onto the Web as an accepted authentication standard for HTTP, it still serves as the basis for all other Web authentication standards.

Under Kerberos the user first authenticates to the Kerberos Domain Controller (KDC). The KDC issues a Ticket Granting Ticket (TGT) to the users Kerberos client. This TGT is then passed to Kerberos enabled applications to let them know that the user has been

authenticated to the KDC. The entire operation is all done using encryption to prevent tampering or sniffing of the network.

Standards for SSO

Because SSO is such a major topic and a wide variety of participants there are currently serious attempts at standardizing SSO in particular for Web applications.

There are at least four major projects in the works, two being under taken primarily by higher education, two by corporations and one being accomplished by mixing both.

WebISO http://middleware.internet2.edu/webiso

The first standard we will look at is the Internet 2 Web Initial Sign-On or WebISO. WebISO is designed to be a specification that defines SSO for Web applications. Because it is coming out of the Internet 2 consortium, it is primarily being defined by higher education institutions and those vendors who serve those communities, similar to how the original Internet was developed.

Currently WebISO has 7 non-interoperable systems that meet WebISO's definition. There is a major thrust to attempt to bring harmony to these systems so that one format can emerge to maximize everyone's resources.

However, one of the implementations, University of Washington's PubCookie has emerged as a leader of the 7 implementations and may very well end up being the 'official' WebISO standard.

Under WebISO a user first attempts to authenticate to a WebISO enabled server. The WebISO application redirects the user to the WebISO login server. After successful authentication to the WebISO server, the user is then redirected back to the original WebISO application and then is let into that protected Web site or application. All of the data is managed using encrypted cookies.

The actual authentication system used by the WebISO login server is not specified it could be LDAP, Kerberos, RADIUS or an UNIX password file.

Shibboleth http://middleware.internet2.edu/shibboleth/

The Shibboleth project is similar to the Project Liberty group (in fact Shibboleth is part of the Project Liberty Alliance – if you can keep that straight, you're ahead in the game).

Shibboleth is primarily focused on inter-organization authorization. That is I have authenticated to my own internal organization and wish to access a resource that exists at another organization. And instead of having to re-authenticate myself to this external organization, I can automagically get access to these remote resources assuming that both my organization and the remote organization have an agreement to allow me to use their resources.

This “agreement” is then enforced using the Shibboleth protocol which exchanges data between parties using the SAML protocol which we will cover later.

It is important to note that Shibboleth is entirely about authorization, not authentication, thus if you want a standard SSO authentication solution you’ll need to implement something like WebISO or Passport.

A true example would a situation like this:

Johnny authenticates to the campus login server, which uses WebISO to transport his authentication credentials to the campus Web applications.

Next Johnny access his online history course which utilizes WebCT, which itself supports WebISO for authentication. His history professor has provided a link to a streaming video clip which is hosted by a Thompson multimedia who published the textbook for his course.

Johnny clicks on the link that takes him to the Thompson site. The Thompson site is protected by Shibboleth. When Johnny clicked on the link to the Thompson site, WebCT (which also understands Shibboleth via its application bridging protocol) passed some data in the URL query string that included a string that can be used to reference Johnny’s campus authorization server (which supports Shibboleth) as well as the URL to the campus authorization service.

Thompson’s server takes the reference string and the URL and then redirects Johnny to his campus authorization service which then sends him back to Thompson. This final redirect contains information (that has been digitally signed) that says that this is Johnny. He’s currently enrolled in his university and in good standing. And that he’s currently online in his history class.

After this bit of magic, the Thompson site then shows Johnny the streaming video clip.

Shibboleth is now in late beta and the first public interoperability display will occur at the October Internet 2002 meeting.

Liberty Alliance http://www.projectliberty.org/

The Liberty Alliance is an organization that was originally founded by Sun Microsystems, Oracle and many other large organizations designed to help develop a standard to share authentication and authorization information between various organizations.

A more cynical observer would say it was Sun and Oracle’s response to Microsoft’s Passport.

Liberty Alliance is designed to follow a mechanism similar to Shibboleth – you authenticate to one system, other systems "trust" the authentication performed at the original site and allows access based on arrangements made out of band.

Again this occurs via SAML and related protocols.
Sun's Identity Server version 6.0 is the first commercial tool to support the Liberty Alliance protocol and Sun has also released an open-source compliance tool to test interoperability with.

Microsoft Passport http://www.passport.net/

Passport is Microsoft's primary attempt at 'standardizing' the Internet authentication space. It was originally designed to be combined with something called Hailstorm to provide centralized management of personal data, primarily for e-commerce applications. Under its current incarnation it is the authentication system for the MSN network and a few e-commerce sites like 1-800-Flowers.com.

Microsoft was not successful at getting many vendors to adopt Passport/Hailstorm because companies do not want a single entity controlling its own user information (you're free to make your own comments on the risks of having Microsoft as that entity in particular).

So now Microsoft has made some comments about federating the authentication under Passport, but there's been no official documentation on the subject.

Passport is already widely deployed, at least within Microsoft's online network. And there are SDKs for Windows, Solaris and Linux (though no "mod_passport" for Apache that I could find). In terms of pure numbers Passport rules the roost but that success has not been able to be translated into success in deployment beyond MSN, but it should be expected that Passport will be the linchpin of the .Net services.

One final note of interest, Microsoft has announced that they wish to make Kerberos the base protocol for Passport in a future release.

WS-Security

http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/

WS-Security is an attempt at standardizing the authentication and authorization services for Web Services. All of the other Web SSO standards have been focused primarily on standard Web browser authentication/authorizations. WS-Security is focused on Web Services (which are XML based RPC protocols), in particular SOAP, which currently has no standard authentication/authorization mechanism outside of any mechanism the transport system may provide (usually HTTP, but you can do Web Services over SMTP or even Jabber IM protocol).

WS-Security was founded by Microsoft and IBM.

SAML http://www.oasis-open.org/committees/security/

Security Assertions Markup Language is an XML format that is used to exchange assertions about a user/entity.

For example an application may assert that "Mark Wilcox was authenticated by password to our LDAP server at 9:45 PM and this is valid for the next 5 minutes". Or "Mark Wilcox is an author of an LDAP book". Or "Mark Wilcox enjoys Mexican food". Or "Mark Wilcox has a valid credit card. You may withdraw 5 dollars in exchange for some nachos".

It uses a combination of standard HTTP posts, Web Services and encryption (XML digital signatures) to pass the data around.

Most (notable exception – Microsoft) vendors who offer authentication/authorization services have adopted SAML.

It recently went out with 1.0 and there have been a couple of inter-operaable events to show support for SAML and it working in 'real-life' settings.

Conclusion

Single Sign-On is a key technology to improving quality of service to users as well as improving the ability to manage a multi-user environment. It is not simply authenticating users but also authorizing their use of a particular application as well.

While everyone pretty much agrees that SSO is a good idea, there are no existing standards that have widely adopted and it may be another 24 months or more before we really see anything emerge.

Thus it pays to be flexible and place your bets on "Show" (be as flexible as possible) instead of picking a particular place on a particular horse.