Extend single sign-on to the cloud

Discover the core concepts of how single sign-on can be extended from your enterprise to the cloud

Kunal Mittal October 19, 2012

Single sign-on (SSO) allows companies to enforce access control policies across multiple applications in a consistent manner. In this article, the author covers the core concepts of how SSO can be extended from your enterprise to the cloud. He discusses several different mechanisms to achieve SSO between traditional applications running in your data center and your SaaS applications. Topics include SAML, OAuth, and traditional agent-based SSO and the author will highlight in general terms about how different Software as a Service (SaaS) providers support the standards to different levels, thus complicating SSO implementations.

Single sign-on (SSO) is an important service that most large enterprises offer to their users — employees, partners, customers, and contractors. In the age of increasing IT security regulations, the use of SSO technologies allows companies to enforce access control policies across multiple applications in a consistent manner, which provides an overall reduced cost of implementation. These policies can include password length, password complexity, password aging, re-use of previous passwords, and more. Whatever the set of rules or policies with which an application must comply, you can implement it once for reuse. Certification and audit are also streamlined for systems that leverage this SSO infrastructure.

Beyond IT compliance, there is significant risk avoidance. How many times have you walked the cubicle isles and seen sticky notes with passwords written on them? What is your personal method of choice for remembering seemingly hundreds of passwords for each enterprise system? Having SSO across your enterprise systems allows your users to remember only one password, reducing not only the risk of having passwords stuck on cubicle walls but also the risk of password sharing. If your email, human resources benefits system, and other systems all use the same password, a user is unlikely to share that password with his or her coworkers.

With respect to cost savings, SSO has proved to have a direct return on investment by a reduction in the number of help desk calls. Fewer different passwords mean fewer times someone will call the help desk for a forgotten password. The savings number quoted in multiple Internet articles and from firms like Gartner and Forrester Research range from 40% to 70% for call volume reduction.
Components of SSO

Let’s start by looking at some of the basic technology components required to support SSO:

- **User.** A user who is trying to log in
- **Web application.** An application that the user is trying to log in to
  For this article, think of the application as any Java™, Microsoft® .NET, or PHP web application or a Software as a Service (SaaS) application such as SalesForce.com, Google Apps, Microsoft Office 365, Concur, ServiceNow, or Workday.
- **Web application agent.** For non-SaaS applications running in the enterprise’s data center, typically installed on the web or application server that hosts that application
- **Policy server/SSO server.** The piece of software that provides all the capabilities and features required to achieve SSO
- **Directory.** The underlying repository that stores the user name, password, and other attributes about a user
  In most organizations, you see Active Directory® Domain Services or some other directory software that implements Lightweight Directory Access Protocol (LDAP). Although not a best practice, you can also use a relational database table.

Figure 1 shows these components in practice.

**Figure 1. High-level components of SSO**

This article focuses on SSO for web-based applications rather than desktop SSO or enterprise SSO. At a fundamental level, web-based SSO principles follow the architecture shown in Figure 1. Let’s make that diagram a bit more concrete by means of an example.

Two key components are required for SSO: The policy/SSO server and the web application agent. The policy server/SSO server is often referred to as an identity decision point (IDP). The IDP is what makes the "decision" on whether the user credentials (user name/password) are correct and whether the user can log in. Every large enterprise software vendor probably provides a piece of technology or a product in this space. Top technologies in this area include IBM® Security Access Manager for Enterprise Single Sign-On, CA SiteMinder, and Oracle Access Management. In addition, many open source and SaaS products are emerging in the market that are becoming serious competitors to those mentioned above — OpenAM, Okta, DirectAxs, and Ping Identity.
Each product mentioned above ships with its own agents, which have to be installed on the web server and application server of the applications you are trying to protect and enable for SSO. In general, you will have agents for most major operating systems, web server software, and application server software. The role of the agent is to intercept the login request for an application, and then pass that request to the SSO server for a decision. Thus, this component is often referred to as the identity enforcement point.

Extending SSO to the cloud

As companies adopt more and more SaaS-based applications, they want to extend their SSO capabilities to those applications. This means a user can log in to an SaaS application with the same credentials he or she uses for corporate applications, email, or even the desktop each morning.

SaaS vendors recognize this need and provide easy mechanisms to help. Many standards have emerged that allow these integrations in a matter of hours or days. This is critical: Imagine if every SaaS vendor provided its own proprietary method for this integration. Even worse, what if each SSO server vendor provided its own technologies? This would certainly lead to a lot of consulting dollars.

SaaS projects are typically business driven. IT does get involved, but sometimes late in the procurement cycle. You have your line-of-business (LOB)-aligned IT executive call the identity management (IDM) program manager saying that the business has chosen a new SaaS vendor and would like SSO. The SaaS vendor claims that the shift is a day's worth of work, so can the IDM team have it done by next week?

You — the IDM program manager — agree to look into it and get back to the LOB executive. You look up the SaaS vendor and read its materials on SSO. You have a quick chat with your architect, and everything looks like it should be a walk in the park.

So, now the technical folks start working on the integration. However, they quickly realize that there are many issues to be worked out and the integration is perhaps a couple of weeks' worth of work — certainly not a day or two. Examples of the kind of issues that can crop up include:

- **Standards are not supported.** In a rare case, you find an SaaS vendor that does not at least support Security Assertion Markup Language (SAML). You more often find that the current version of your SSO server may not support SAML or your vendor has an SAML pack as a separate SKU. So, you now have a procurement cycle to deal with.

- **A standard is supported but does not work.** Assuming that you got past the first issue, it's also probable that you see that even though both ends (SSO server and SaaS vendor) claim to fully support a particular version of a specification, when you try to integrate them, you run into some challenges. These challenges are generally minor and can be worked through, but they require that you get both support teams, along with your IDM technical folks, on a conference call.

- **Proprietary technologies.** Some SaaS vendors provide web services or directory synchronization as an alternate mechanism for SSO. Although viable, these alternatives lead
to additional development effort, and having to deal with network security, firewall, and other issues that make the project take longer.

The point is clear: Set expectations carefully. An SaaS SSO project is not a 24- to 48-hour task. The actual work effort may not be more than that, but the overall time could easily be a couple of weeks because of all the coordination required.

Design patterns

This section discusses three design approaches to achieving SSO between your traditional data center-hosted applications and the SaaS applications. You have undoubtedly seen patterns implemented that take into account the challenges already mentioned.

Design pattern 1: Custom web application

You can use this design pattern if your SaaS provider or SSO software doesn't support SAML. In addition, use it to consume web services or another proprietary integration. A final use case for this design pattern is when your SaaS provider supports SAML but your SSO server does not. If you end up deciding to do a custom SAML implementation (not recommend), the design pattern will be similar to the one described here.

Essentially, you are going to deploy a web application protected by a typical web agent that your SSO software provides. This application contains a login page and authenticates against your SSO server. The application also contains code to pass the credentials to the SaaS application. After the user is authenticated, the application passes authentication information to the SaaS application. Of course, always encrypt this data and transmit it over Secure Sockets Layer. You can use a custom SAML implementation to generate SAML if the SaaS provider supports SAML, or, alternatively, the application can have the code to consume a Web service that the SaaS vendor provides. A least-preferred but viable technical option is to use an HTTP post with some basic information that allows the SaaS application to recognize the user.

A variation to this pattern is that the SaaS provider actually provides this web application. Essentially, this web application is then no different than the agent you're installing from the SSO server product and so is probably a preferred option to building a custom web application.

Design pattern 2: SAML based

This is the ideal solution, where SAML is used to transmit the SSO information between the SaaS application and the SSO server. If both sides fully support SAML, an SAML setup typically takes a few hours to perform and test.

This is where the different SSO server products are still maturing. With every product cycle, you start to see more out-of-the-box functionality that reduces the integration effort to sometimes as little as 5 minutes. For example, I integrated SalesForce.com, Google Apps, and Facebook with one SSO server product in an hour. As illustrated in Figure 2, here’s how this pattern works:

1. The user logs in to an SaaS application.
2. The request is redirected to your IDP (SSO server) in the form of an SAML request.
3. Your SSO server decrypts the SAML and authenticates the user.
4. The server returns a success or failure SAML response to the SaaS application.
5. The user is logged in or shown an appropriate error message.

To set this process up, the generic steps are as follows:

1. Log in to your SaaS provider's admin console, and define the URL to your SSO server with credentials.
2. Set up the SSL keys for the encrypted messages on both the SaaS server and your SSO server.
3. Create a policy on your SSO server, and enable it for SAML communication.
4. Tweak any other parameters required, such as custom login and error pages and timeouts.

If all goes well, it really is that simple.

**Figure 2. SAML-based SSO**

**Design pattern 3: Automated provisioning**

Automated provisioning is a somewhat fancy alternative to a directory synchronization design pattern. **Figure 3** demonstrates how it works. During the on-boarding, transfer, and termination processes, as an IDM tool writes changes to the user identity to its own data store (typically directory software), it also writes this information into the SaaS application, typically via a web service call. User additions, termination, and password changes are all sent to the SaaS application and are thus kept in sync with the internal directory.

Now, when a user logs in, he or she can use corporate credentials. This setup achieves a slightly different flavor of SSO called *simple sign-on* instead of SSO — simple, because it still uses the same user name and password as the corporate system but is not single sign-on. The user does not have to log in separately to the SaaS application.

You can implement this pattern via custom web services implementations or if the technologies being used support the Service Provisioning Markup Language (SPML) standard for this type of user provisioning activity.
In conclusion

Establishing SSO with SaaS providers is a relatively easy task if well planned out. There are some things to watch out for, and the whole process requires good project management and expectation-setting skills to ensure that your project is successful. Remember that SaaS vendors support SSO, but not all support SAML consistently. Therefore, it's important to have technical conversations about SSO at the beginning of the planning or procurement cycle. Two-way sharing of technology limitations on both ends is required.

Similarly, if custom solutions are being built, ensure that they are highly secure and encrypted. In addition, ensure that you log everything. If and when things go wrong or concerns arise that a login transaction is taking too long, you want to trust your logs. A detailed logging approach can generally pay back the time spent on the first support issue.

Finally, if you see SaaS playing a more prominent role in your IT landscape, plan ahead. Make the right technology investments sooner rather than later. It is also important to plan to refactor and reevaluate as SaaS vendors and commercial off-the-shelf products evolve their offerings.
Related topics

- Wikipedia provides good background on SSO concepts and related IDM concepts.
- Read more about SAML and the role it plays in SaaS SSO.
- Learn more about OAuth technology and how to use it.
- Learn more about CA SiteMinder.
- Learn more about ForgeRock OpenAM.
- Learn more about the open source policy/SSO server options for the cloud mentioned in this article:
  - ForgeRock OpenAM
  - Okta
  - DirectAxs
  - Ping Identity
- In the developerWorks cloud developer resources, discover and share knowledge and experience of application and services developers building their projects for cloud deployment.

© Copyright IBM Corporation 2012
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)