Single Sign On – A Contrarian View

Keys Botzum
Senior Consulting I/T Specialist
IBM Software Services for WebSphere
August 6, 2001

Summary

This document outlines some of the issues associated with Single Sign On (SSO) in an enterprise environment. Many organizations embark on these efforts without thoroughly considering the issues, costs and benefits involved. This article will help you make an informed decision about deploying SSO.

The views expressed in this document are those of the author do not represent any official IBM position.

I. Introduction

SSO is the holy grail of many organizations. With SSO, users will log in once to an SSO domain and then are never challenged again while accessing secured resources within that domain. The Open Group defines SSO as:

> *a mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where that user has access permission, without the need to enter multiple passwords.*[1]

Originally, SSO was to be achieved by developing all applications and tools to use a common security infrastructure with a common format for authentication information, avoiding the current situation of heterogeneous security infrastructures. This approach has a number of valuable benefits:

- For end users:
  - Only one authentication mechanism to remember. For password-based authentication, this means only one password to remember and update, and one set of password rules.
  - A single sign-on for each user into the SSO domain, typically once per day.
- For Operations:[2]
  - A single common registry of user information (possibly replicated)
  - A single common way to manage user information
  - A single common security infrastructure
- Security advantages:

[1] Taken from the Open Group web site (http://www.opengroup.org/security/topics.htm) from the security section.
[2] This amorphous group is the set of people that run the computer systems in this organization on a day-to-day basis.

- A common secure infrastructure leveraged enterprise-wide, that can be carefully managed and secured
- Depending on the security approach, secure delegation of credentials is possible. This enables end-to-end security, possibly across application and system boundaries.
- Easier to manage and protect common registry
- Easier to verify user security information and update when necessary rather than tracking down all operational systems. This is particularly valuable when users move to new roles with different access levels.
- Common enterprise-wide password and security policies.
- Users less likely to write down passwords since they only have to remember one

II. The Problem

Creating a common enterprise security infrastructure to replace a heterogeneous infrastructure is without question the best technical approach. This is being attempted with technologies like the OSF Distributed Computing Environment (DCE), Kerberos, and with PKI-based systems, but few, if any, enterprises have actually achieved this.

Unfortunately, the task of changing all existing applications to use a common security infrastructure is very difficult. This has been further hampered by the lack of consensus on a common security infrastructure. Today we have many proprietary security systems as well as several competing security standards and pseudo-standards: Kerberos, DCE, Microsoft ActiveDirectory and related technologies, PKI, etc. Of course, these technologies do not interoperate seamlessly. As a result, these proprietary and standards-based solutions, while very appropriate, cannot be applied to every system. For example, until recently,[3] WebSphere Application Server could not recognize authentication information from other. Problems like these occur over and over again across an enterprise as organizations buy products without fully considering the implications.

III. The Proxy-SSO "Solution"

Since moving all applications to a single security infrastructure is extraordinarily difficult, many enterprises attempt to create the illusion of one for end users by using SSO products that authenticate to existing legacy systems without user intervention. I refer to this as *proxy-SSO* to distinguish it from "true" SSO achieved by creating a common security infrastructure. These products typically use some proprietary authentication mechanism and then reauthenticate transparently to multiple underlying systems via the user interfaces to those systems. This involves a scripting engine that drives the interaction with each underlying system's authentication challenge. The scripting engine simulates a user logging in, so the underlying systems do not need to be changed. A proxy-SSO database stores all combinations of user names, passwords, and systems.

[3] WebSphere Application Server 3.5.3 or later supports a feature known as Trusted Associations that allows its security infrastructure to recognize (via custom coding) credentials from other security systems when provided to Web-based applications.

IV. The Solution is Difficult, Expensive, and Incomplete

The task of configuring a proxy-SSO system for a large enterprise with numerous existing systems is staggering, as custom scripts must be developed for every possible user authentication interaction. Just considering Unix utilities, we have: ftp, telnet, rsh, and rlogin. Add to this list the utilities for other operating systems, custom applications, and Web sites, and the number is enormous. In addition, once the scripts are created, they must be maintained. As the underlying systems change their interfaces, the scripts require changes. This results in an ongoing maintenance cost that is rarely considered. The scripting also hides the "front-door" to applications from the end users and thus might bypass important information on the login page such as security warnings.

A central proxy-SSO database of username and password mappings must be also be created and maintained. This database must follow all password rules for all enterprise systems. It must be updated whenever underlying passwords are changed, or it must be the only system that changes passwords (creating yet another scripting task for each enterprise system). This doesn't even address systems that use other authentication mechanisms such as certificates, challenge-response, SecureID cards, etc.

Here is a summary of the costs of proxy-SSO:

- Initial costs
  - Product purchase
  - Customization of product for existing systems. This usually involves creating custom scripts to drive the legacy systems. This is a large work effort.
  - Loading existing user information into the proxy-SSO solution and deploying to users.
- Ongoing costs
  - The usual software upgrade costs
  - Another registry to maintain. This system must be highly available.
  - Password management by users. When passwords expire in the various legacy systems, users must update the legacy system and the SSO system. Since users are no longer familiar with the login procedure for the legacy systems, they may not even know their passwords[4]. This may make it impossible for users to change their own passwords without assistance.
  - Script maintenance. As the legacy system user interfaces change, the scripts have to change. A significant work effort.

Many of these costs can be controlled by carefully designing the underlying applications to work with a script-based proxy-SSO solution, but this largely negates the main advantage of these proxy-SSO solutions: no need to modify legacy systems.

[4] This can be alleviated by two approaches: 1) the proxy-SSO product can change passwords automatically although this will require still more scripting and a more complex product that understands the password rules for each underlying system, can detect when passwords are no longer valid, and can script the changing of passwords; 2) the proxied systems can move to fixed, never changing passwords. This 2nd approach is only feasible if the proxy is the only way to access the existing systems. Ensuring that raises a set of issues that are beyond the scope of this article.

On the benefit side, the proxy-SSO solution saves users the trouble of typing in their user name and password to several different systems several times per day and remembering all of these username and password combinations. While this is certainly an issue, it is not clear how much real benefit this provides to the organization. The most troublesome aspect of multiple authentication systems is probably remembering the number of passwords, not the actual authentication effort. This point will be revisited later. Of course, in high-cost situations, such as call centers, these benefits are significant and can probably justify the cost of a SSO solution. In other, more routine environments, the benefit is far less clear.

From a security perspective, the results are mixed. The enterprise still has multiple incompatible security infrastructures and now has one more security product and one more user registry (another point of attack). On the positive side, if the proxy-SSO solution can manage the changing of user passwords in all legacy systems transparently, then users will no longer have difficulty remembering passwords and will no longer write them down.

In summary, proxy-SSO solutions provide limited benefit to end users, do not address most of the operational and security costs of a heterogeneous security infrastructure, and introduce new operational costs.

V. The Web

Recently, efforts to achieve full SSO have largely died out. A newer form of limited SSO is becoming popular: Web-based SSO, in which products authenticate users using some mechanism and then create credentials that can be used by downstream Web applications for authorization[5]. Unfortunately, as in the non-Web space, changing all existing Web systems is difficult. Thus, people are again embarking on proxy[6]-SSO solutions for the Web. This is considered simpler because Web applications have just one interface: the Web browser. However, while a single user interface simplifies the problem, it does not mean that all Web applications authenticate the same way. There are many different ways of asking a user to authenticate. Even in a simple Web based application, the proxy-SSO scripting engine must be able to traverse an arbitrary Web application and enter input just as if the user had done so. This problem, while significantly simpler than enterprise proxy-SSO is still difficult.

To script a Web interaction around a password-based authentication system, the following issues must be considered:

- The initial request to the target site must be rerouted to the proxy-SSO server that will then retarget the request appropriately.
- Basic authentication is trivial because the challenge protocol is well-defined and easy to intercept.

[5] IBM's SecureWay Policy Director is an example of a product that does provides Web-based SSO, although Policy Director provides far more than just that.
[6] In this context the word proxy has nothing to do with a Web proxy server.

- Form-based authentication presents the following unique problems:
    - The login page must be loaded and parsed in order to extract the hidden fields and find the action and target servlet
    - Forms may ask additional questions beyond user name and password
    - There may be more than one action on the form
    - Sites may use elaborate JavaScript to challenge the user and/or encrypt the user's authentication information before sending it back to the server.
    - Browsers are notoriously lenient in their presentation of illegal HTML. The proxy-SSO solution needs to work with illegal but displayable HTML to avoid forcing changes to existing sites.
- Traversing a Web site to simulate a user login interaction requires addressing these issues:
    - Some sites present informative messages to users before the login process starts. Bypassing these messages, particularly if they are security warnings, is problematic.
    - Cookies sent before the post-login welcome page of the target application must be preserved.
    - Sites that detect the browser type and make decisions before or during the login process must be handled. This information must be passed through from the browser by the proxy-SSO solution.
    - Sites may create dynamic URL strings during and before the login process with encoded information in the URL (for example, http://www.abc.com/index.html?session=12345).
    - Sites may pop up additional browser windows during the login process
- Proxy server (in the Web proxy server sense) solutions must address these additional issues:
    - URLs embedded in the responses must be rewritten to point to the proxy. Embedded URLs may be absolute (for example, http://www.abc.com/app/page.html), server root relative (for example, /app/page.html), or page relative (for example, page.html).
    - URLs embedded in JavaScript must be rewritten. For some systems, this may be impossible if the URLs are computed on the client. Even if not, this requires parsing the entire response stream looking for particular characters. This will affect performance.
    - Cookies sent back by the site must have the domain and path values rewritten to point to the proxy so the browser will not discard them.
    - Multiple sites, when combined under a single proxy, may reference resources that conflict with each other. URLs and cookie names may no longer be unique.

Some applications that appear to be Web applications are sufficiently complex that they are really full-fledged applications. Many Web applications use plug-ins, extensive JavaScript, and Java applets to create a fuller user experience. These types of applications may not be addressable via the techniques described above.

Web application user interfaces change rapidly and this will affect scripts that depend on the existing interface. While the Web solution simplifies the client interaction problems, it is hardly trivial. Finally, the other operational issues (including heterogeneous infrastructure, multiple, registries, and password maintenance) associated with a proxy-SSO solution remain.

VI. An Alternative Approach

We've discussed the weaknesses of the proxy-SSO approaches. Now it is time to try to determine an approach with as many of the benefits of "true" SSO as possible without new operational problems.

Looking back at the benefits of SSO from section I, we can see that most of the benefits derive from the single registry. Here is the list again, this time with items highlighted in bold that are the direct result of a common registry:

- For end users:
  - Only one authentication mechanism to remember**. For password-based authentication, this means only one password to remember and update, and one set of password rules.**
  - A single sign-on for each user into the SSO domain, typically once per day.
- For Operations:
  - **A single common registry of user information (possibly replicated)**
  - **A single common way to manage user information**
  - A single common security infrastructure
- Security advantages:
  - A common secure infrastructure leveraged enterprise wide that can be carefully managed and secured
  - Depending on the security approach, secure delegation of credentials is possible. This enablse end-to-end security, possibly across application and system boundaries.
  - **Easier to manage and protect common registry**
  - **Easier to verify user security information and update when necessary rather than tracking down all operational systems. This is particularly valuable when users move to new roles with different access levels.**
  - **Common enterprise-wide password** and security **policies**.
  - **Users less likely to write down passwords since they only have to remember one**

From a business perspective, the items in bold are the ones with the greatest benefit. While a complete solution is most desirable, this comes remarkably close. While end users get most of the benefits of SSO with a single registry, an organization can also significantly reduce the cost of registry maintenance simply by consolidating to single

registry. End users must authenticate multiple times per day using the same username and password, which, while irritating, is not a significant cost.[7]

It's worth nothing that a proxy-SSO solution can be built on top of a common registry solution. This will greatly simplify the proxy-SSO problem and result in a better overall infrastructure. However, it is not clear if this is worth the additional cost given the limited benefit.

Of course, there is no panacea. Using a common registry requires that all existing applications be migrated to use this new registry and that some organization must manage this critical business system. While not trivial, this cost will be leveraged across numerous applications that no longer have to manage a user registry.

The best registry to choose is one that supports Lightweight Directory Access Protocol (LDAP). There are a number of LDAP server products,[8] and most security and middleware products that support secure access can use an LDAP directory for authentication. Once an enterprise embarks on this effort, they may find that many applications can switch easily to LDAP simply by reconfiguring the middleware that they use. For example, Netscape Enterprise Server, Apache, IBM HTTP Server, AIX, Solaris, and WebSphere Application Server already support LDAP out of the box. Additionally, Microsoft ActiveDirectory can provide LDAP services to other clients.

How to select an LDAP directory product is beyond the scope of this article. As with any enterprise product, you should examine vendor support, scalability, performance, replication, fault tolerance, extensibility, security, tools, and features. In particular, consider password management features, such as account lockout and strength rules. These are not part of the LDAP standard, so they will be vendor-specific.

For applications that use custom security registries,[9] change will be required. Fortunately, LDAP APIs are available for many programming languages, including C/C++, Java, and Perl. The APIs are quite simple so conversion of existing applications is feasible. While it is not realistic convert all existing applications, it is reasonable to expect new applications to use a common registry. Additionally, most PKI-based systems require a registry for user information, and LDAP is widely supported. Since many organizations are moving toward PKI anyway, this can be viewed as a stepping-stone toward a full PKI-based security infrastructure. Existing applications can be converted to use the LDAP registry as appropriate based on cost and benefit. Over time, this will greatly improve the situation without introducing a large and complex system that requires ongoing maintenance without significantly improving the situation. For those considering an SSO solution now is the time to act. With each passing day, more and more incompatible systems are being deployed in your enterprise.

---

[7] As before, in certain situations, such as call center applications, the cost of signing in repeatedly is significant.

[8] IBM bundles our SecureWay LDAP Directory with many of our products, including WAS.

[9] This alone should raise concerns. If an application team develops the registry, how does the enterprise security team know that the registry is well designed and well managed?

VII. Conclusion

The paper has discussed the costs and benefits of two approaches to SSO: a homogenous infrastructure and proxy-SSO. A practical alternative to SSO involving a common registry has also been discussed. The "true" SSO approach via a homogenous infrastructure and the alternative common registry approach are superior to the proxy-SSO approach in most situations.

You should now more fully understand the issues, costs, and benefits surrounding SSO solutions. Analyzing the costs and benefits for your own organization requires careful analysis and experimentation by highly skilled people. Should you then decide to go forward with a proxy-SSO solution, you will at least understand the task that lies before you.

VIII. To Learn More

Below is a list of SSO products ("true" SSO and proxy-SSO) as well as some other views on SSO.

- An InfoWorld article that claims that proxy-SSO is just great: http://www.infoworld.com/articles/es/xml/00/10/02/001002esnsso.xml
- Open Group security Web site includes discussions of SSO: http://www.opengroup.org/security/topics.htm
- IBM Policy Director, a product that supports SSO and a variety of other security features: http://www.tivoli.com/products/index/secureway_policy_dir/index.html.

Additional Information:

- Reader unfamiliar with the terminology in this paper may wish to read Enterprise Application Security, by Keys Botzum – http://www.pittsburgh.ibm.com/~keys. It is in the documents section.
- *Understanding and Deploying LDAP Directory Services*, Howes, Smith, and Good, Macmillan Technical Publishing, ISBN 1-57870-070-1.
- The HTTP Cookie specification, RFC #2965, available at http://www.ietf.org.
- IBM SecureWay LDAP Directory: http://www.ibm.com/software/network/directory/
- Iplanet LDAP Directory: http://www.iplanet.com/products/iplanet_directory
- *Understanding LDAP* (SG24-4986-00), IBM Corporation, International Technical Support Organization http://www.redbooks.ibm.com