Secure Coding.
Practical steps to defend your web apps.

Secure implementation of Enterprise single sign-on product in an organization

v.1.4b

Ravikanth Ponnapalli
July 14, 2004

Contents

1. Abstract

Single Sign-On is a very important component of the security architecture of an organization. In IT, it is generally believed that it is expensive to deploy an enterprise Single Sign-On solution that is secure and scalable. However, there is a growing awareness in IT management about the advantages of implementation of enterprise Single Sign-On. It can be overwhelming for project managers to understand what it takes to build a secure Single Sign-On solution and whether it is viable to build the solution in-house or to buy a product and implement it. The topic - “buy versus build” is not discussed in this paper. There are a lot of resources on the Internet that can help to educate you on this topic. The references section at the end of this paper provides a link (reference #3) to one such resource. Even after voting to “buy”, it can take significant efforts to evaluate the products and choose the SSO product that is suitable for the organization. This paper aims to usher IT managers, developers and other resources interested in such a project by providing a blue print of the effort needed to implement an enterprise Single Sign-On solution with security in mind.

2. Overview of Single Sign-On

2.1 What is Single Sign-on?

We will go through a brief introduction to Single Sign-on (hereafter also referred to as SSO in this document). SSO can be defined as a user experience of logging in just once and being able to navigate across many applications seamlessly without a need to enter credentials for each application. It is very common for organizations to have many applications running to take care of different business functions. SSO makes it easy for the users to login once and be able to access all the applications they can, reducing the need for users to remember a plethora of logins and passwords. The following is a brief description of a few important concepts of SSO.

Authentication

The process of verifying the user’s identity, making sure that the user is who he claims to be. This can be based on login & password combination or Smart card, biometrics, etc.

Authorization

The process of verifying whether a user is privileged to access a particular resource.

Credentials

Credentials are the details provided by a user during the process of authentication into an application. They can be login and password, fingerprint, smart card etc.

Domain
A domain is a logical group in an organization with a unique name that is the part of host names used on the intranet/Internet. For example, mycompany.com is the domain name in myhost.mycompany.com where as mycompanystore.com is the domain name in www.mycompanystore.com. While mycompany.com is a parent domain, it.mycompany.com is a sub domain reserved for the IT department in the organization.

Protected Resource
It is a resource the access of which is not open to everyone. A user needs to go through authentication and authorization before accessing a protected resource. It can be a URL on the Internet or intranet, a client to an application, a folder on a server, etc.

Cookie
A cookie is a ticket given to a user's browser as a result of successful authentication and it contains data to indicate authentication and authorization information. The actual contents of a cookie may vary depending on the application. After having a cookie, if the user browses to a different application that is a part of SSO, the cookie is presented by the browser to the application in lieu of credentials, for directly logging into the application. Then if the user is authorized to access the resource, he will be able to do so. There are two types of cookies.

- Per session cookie: The cookie is retained in volatile memory of the user's computer and is valid only till the end of the current session. As soon as browser is closed, the cookie is destroyed.
- Persistent cookie: When you check the box for "Remember my password" or "Log me in automatically" or a similar one in the login page of a web site then a cookie is stored on your hard drive. This is a persistent cookie and the server that sets the cookie specifies its life. So, its contents are not destroyed when the browser is closed and will remain available between sessions, till expiry.

Very good resources are available on the Internet to provide a detailed description on the basics of SSO. A few of them are listed in the references section of this paper.

2.2 Different types of SSO
Most of the SSO products available in the market can be categorized into two types based on the architecture.

- Web-based (also known as enterprise SSO or ESSO) and
- Non Web-based (also known as legacy SSO).

The focus of this paper is ESSO. For the benefit of readers, there is a brief introduction to legacy SSO given below.

Web-based SSO can be categorized as follows.

Internet facing: This generally takes care of the SSO for the applications that interact with customers. For example, an organization may have

- A customer support application that lets customers login, raise service tickets and browse through the knowledge base and
- Another application that lets customers view their bills online and pay them online.

These applications can be knit using SSO so that customers can login only once to be able to access both the applications.

Intranet facing: Generally this takes care of the SSO for the intranet applications that internal users of an organization interact with, using a web browser. For example, an organization may have

- An expense reporting application that lets employees submit their expense reports, track their status online and
- Another application that lets the users plan their upcoming travel needs.

These applications can be knit using SSO so that the user can login only once to be able to access both the applications.

Multi-domain: This can be categorized into two types.

- SSO between two or more different domains in the same organization (often referred to as Intra-organization multi-domain SSO). For example, an organization may store its confidential marketing content on content.mycompanymarketing.com and available opportunities on opportunities.mycompanypartners.com. These applications can be knit using SSO so that the partner user can login only once to be able to access both the applications.
- SSO between the applications of two or more organizations, usually partners (often referred to as cross-organization SSO). A new concept called federated identity management emanated recently to improve security in the architecture of cross-organization SSO. We are going to discuss it in detail later, in the architecture section of this paper.

Most of the products available in the market support all these categories of enterprise SSO. We will take a closer look at each type of deployment in the architecture section.

Legacy SSO products use a different approach to SSO. A component called the SSO agent gets activated when a user logs into his workstation and remembers the logins and passwords of the user to different applications. It provides the credentials to the applications when the user tries to log into these applications subsequently. Thus, with one authentication session initiated by the SSO agent, legacy single sign-on enables user navigation to various applications on an intranet. Recent versions of legacy single sign-on products support smart card, PKI, and biometric authentication.

A few SSO products available in the market are given in the references section.

2.3 How can SSO products address security issues in an organization?

The marketing literature of available products in the market can be misleading to get managers to believe that an SSO product is a cure-all. SSO products come with many features that augment the security in the infrastructure of an enterprise. However, it is a fact that any SSO product cannot resolve all the security concerns of an organization. Without very careful planning, implementation and verification, SSO products can introduce new security holes. By and large, the following are some of the ways in which SSO products can add security to infrastructure.

- Reduces the need for users to remember many logins & passwords and hence reduces the chance of users following insecure practices like writing the passwords on post-it slips pasted to their workstations etc.
- Central enforcement and administration of security policies on the applications participating in SSO.
- Enforcing restrictions on passwords like minimum length, having at least one special character, character in upper case, numeric digit, etc.
- Forgotten password recovery mechanism can alleviate a user from the fear of forgetting passwords. Users may otherwise follow insecure practices, as mentioned above, to make sure that they don't forget passwords. However, it is important to keep in mind that the password recovery mechanism, like a security question and answer, can open up new security issues.
- User lockout after a few consecutive unsuccessful attempts.
- Enforcing change of password on first login or after password reset by an administrator.
- Maintaining password history and making sure that a new password doesn't match with a password in the history.
- Expiring passwords after a specified amount of time.
- Session timeout. A user session can be timed out because the session has been inactive for a specified amount of time (called inactivity session timeout) or because the user session has been in progress continuously for a specified amount of time say, the last 2 hours (called absolute timeout).
- Advanced monitoring and reporting capabilities.

Some of the features mentioned above might already exist in your organization in one or more applications. Implementation of SSO product can be helpful to homogenize these features across all applications of SSO in the organization.

3. Life-cycle of a Single Sign-On product implementation

3.1 Evaluation of available products in the market

> When you look for an enterprise solution that increases security and usability, SSO is a key piece of the puzzle. When you evaluate SSO solutions, you need a product that truly provides single sign-on to all of your applications without using risky and costly integration techniques (e.g., scripting or agents). SSO should be easy-to-use and manage, robust, reliable, secure and ready-to-scale to meet your future needs. You need an SSO solution that will work with your existing and future infrastructure, including your chosen authenticator and directory service (Boroditsky, p.7).

So, a careful evaluation is essential before choosing to implement an SSO product. Evaluation can be divided into the following steps.

3.1.1 Functional evaluation

This is the place where we start understanding what the product can do by analyzing the functional features of the SSO product. Developing a prototype is a commonly followed approach that can help improve visualization of the functionality of the product. A recommended way to begin functional evaluation is to come up with a set of basic requirements after discussions with representatives of different applications. You can categorize the requirements as "must" and "nice to have". A prototype can be made available in a development environment with development versions of the deployed applications tied together using SSO. It can then be analyzed to understand how well each product under evaluation meets or exceeds the requirements. Features like authentication, authorization, Password restriction enforcement, etc. can be evaluated using the functional evaluation.

3.1.2 Security and usability evaluation

It is a common observation that introduction of new security measures can reduce the usability of applications. Striking a reasonable trade-off between security and usability can get tricky and hence needs involvement of resources from different departments.

> SSO cannot achieve the objectives on its own. In order to overcome challenges in user experience and security barriers, security and usability professionals need to work more closely together. Both are disciplines that require special consideration. Both security and usability cannot just be added at the last moment. They must be fundamental to an application's design, built into every feature, and considered in each modification. Both require rigorous testing and constant review (Boroditsky, p.5).

Involving representatives of user community in evaluation can help in analyzing how usable the features of the products are. For example, an end user can

explain how he feels about the new restrictions and policies on password management, a system administrator or a resource from IT security can evaluate how good the monitoring and reporting tools provided with the product are, while an administrator can explain on how easy or tough is the administration of the product. Since the product brought in is going to directly deal with security, it is needless to mention that meticulous care needs to be taken in analyzing how the SSO product can augment the already existing security infrastructure without introducing new vulnerabilities. Following are some of the questions that can help in an effective security evaluation of the product.

1. What are the different authentication methods that the product can support? If login & password are used, is form-based authentication supported? What happens behind the scenes in a user login scenario? How tough or easy is it for a user to sneak into the application, bypassing authentication?

2. Where do the components of the application reside in the architecture? If there is a need for SSO of multi-domain applications, how is the architecture affected and what are the built-in features to ensure security of the architecture?

3. What are the communication protocols they use? Is the communication encrypted or not? If encrypted then how secure is the algorithm used?

4. If cookies are used then what details are stored in the cookie and if encrypted, what is the method used for encoding/encryption? Can a malicious user exploit the known (or unknown) vulnerabilities either on client or on the server to read or spoof the contents in the cookie? Is there a way to enforce inactivity or absolute session timeouts?

5. Are the passwords stored in an encrypted form in the data repository? Is there any temporary storage for passwords that needs to be used either during migration from one environment to another or in any other situation? Are the passwords stored in clear text or stored in an easy-to-break form (may be encoded with trivial algorithms like Base 64) in any context of the product?

6. What is the data that is cached in different components of the product? Can there be any possible security holes because of this caching? Is there a way users can access the data that is cached? Are the credentials of the user cached? If so, how easy is it for an intruder to access this data? What are the precautions to be taken in this context?

7. What are the log levels that the application can have and what is the implication of each log-level? How frequently do the log files need to be purged from server(s) to make sure that the disk is not full? Will the logins

and passwords be dumped into the log file in clear text in any of those log levels?

8. What are the different modes of administration that are supported by the product (like an installed admin console, web-based admin console, etc)? Are there any known vulnerabilities against the administration component of the product? It is important to understand that administration components are generally the hot targets of intruders.

9. What are the recommended configurations based on experience of deployments at other customers?

10. What is the effort needed to migrate the configuration in one environment to another environment like migration from development environment to QA or QA to production etc?

11. What is the vendor's Service Level Agreement (SLA) for fixing any security holes discovered after the product gets implemented? How does the vendor inform customers when such security patches are available? Is it through email or web-based bulletin or discussion forum etc?

It is always a good idea to involve a resource from the IT security team from the beginning of the evaluation process. After choosing a product, this resource can be involved with the entire implementation to ensure that the implementation is secure in all aspects.

3.1.3 Scalability evaluation

Every organization tries to grow in the course of time and so does IT infrastructure by taking up new projects. For an SSO product scalability is an important aspect of evaluation. Decision makers responsible for the implementation of the SSO product like project managers need to have a vision on the possible direction of growth for the organization and how the SSO product can help by scaling up to the needs of future applications. Following are some of the questions that can help in this evaluation.

1. How many concurrent users can the product support now? How many concurrent users do you expect to have after the implementation of the product? Does it need new licenses when we add new user base? What are the cost implications of these new licenses?

2. What are the possible future applications for SSO that may be developed in the short and long term? How do these applications affect the number of concurrent users? Will the SSO product be able to scale to that user count?

3. What is the caution that needs to be taken while designing the architecture of the implementation to make sure that it is scalable for future applications? For

example, does the product support adding more servers with increased user load and configuring to have load balancing among the servers? Or does the product only support a primary and secondary configuration? How easy or difficult is it to change from one type of configuration to another?

4. How easy or difficult is it to upgrade hardware after the SSO is implemented with the product? Are there any known issues here?

3.1.4 Integration evaluation

In SSO product deployments, integration with enterprise and/or other applications is the traditional and most likely point of failure. Hence, if there are different enterprise applications running (like ERP or CRM applications etc) then the ability for the SSO product to glue these applications together to allow SSO in navigating between those applications can be one of the foci of evaluation. Some of the SSO products are already certified with the vendors of these enterprise applications. If integration with enterprise applications is high on the list of priorities then it can be easier to consider for evaluation only those SSO products that are certified with the vendors of those enterprise applications. An important point to remember is that certification of an SSO vendor with the product of an enterprise application vendor doesn't necessarily mean that the integration will be smooth and secure. Integration can still mean a lot of effort and it can take a long time before security issues are identified after an SSO product got certified with the vendor of an enterprise product. It can get tricky here. The longer it is since the certification, the higher is the probability that a majority of these issues are identified and hence more secure the integration can be. For fixing the identified security issues after certification, you need to apply the patches from the vendor either individually or in the form of a maintenance release and these patches can open up a new set of incompatibilities or security issues. Making sure to evaluate the exact version of the SSO product that got certified is probably a safer approach from integration point of view. The prototyping technique discussed in functional evaluation can be handy in integration evaluation in analyzing such aspects. Following are some of the questions that can help in this evaluation.

1. What is the effort involved in integrating the SSO product with enterprise applications? In other words, how easy or tough? What is a rough estimate on the average number of person hours taken for a similar integration in prior implementations?

2. Is there an existing directory server? If not, what are the security implications of introducing a new directory server? What is the secure configuration for a new directory server's access control lists?

3. Are there any known security holes that are created as a result of such integration in the version of the product?

4. What is the latest version of the SSO product that got certified with the enterprise application? What is the roadmap for certification of next version? When was the last security audit (internal or third party) on the SSO product and on which version of the product?

3.1.5 General evaluation

The evaluations mentioned so far are technical in nature. They can be complemented with a generic evaluation of the SSO product vendor. Following are some of the questions that can help in this evaluation.

1. How long has the company been in business?

2. How many customers does the vendor have? How many of those have successfully implemented SSO into one or more of their customer-facing applications?

3. What are the minimum hardware and software requirements for implementing this product? How much does it cost to procure them?

4. Are there any known vulnerabilities in the product? If so, how quickly did the vendor respond to those vulnerabilities with a patch?

5. Does the product go through third party security audits? If so, how often?

3.2 Planning the Implementation

As we all know, things go better with a plan. Careful planning with different dimensions of the challenges in mind can pave the way to a successful implementation of any project. The following gives an overview of some important criteria while planning the implementation of SSO.

- Plan for training of resources on the chosen SSO product. Note that there can be different types of training targeted at end user, developer, system administrator, business analyst, etc. How many resources need to be trained? What is a typical skill set needed for a resource to start working on the implementation? What is the security awareness they have?
- If you don't have the needed expertise in-house to start implementation, plan for help from a consulting firm that is well experienced in such implementations. This way, in-house resources work by the side of experienced consultants in observing critical aspects of implementation and learn from their experience.
- Set expectations of business and user community in the possible features and security that they can expect out of the SSO product after implementation. As mentioned before, SSO can be commonly misinterpreted as a cure-all for security. Setting realistic expectations avoids any surprises and the resulting frustration in business and end users.

- Start simple. Start with implementing SSO between two of the simplest of the applications you have. Since all the architectural foundations are laid out for the initial deployment, a successful implementation will not only set up the needed infrastructure but also will increase the confidence and morale of the associated resources. Expand your SSO to other applications after a satisfactory initial deployment.
- Plan for sufficient cushion time to absorb any unexpected delays. If the project timeline slips in the initial deployment it can be very demoralizing for the user community and the business representatives along with other groups that are affected by this. Discuss with people who are experienced in implementing the SSO product to understand the magnitude of the cushion time at different junctures in the project.
- If possible, test usability of SSO just as a marketing campaign is tested: first, a focus group; second, a pilot; third, an expanded pilot with multiple locations; and fourth, deployment (Boroditsky, p.6).
- Take into account input from the most inexperienced users. The number of novices (less than 18 months of experience) should be greater than the number of veterans (Boroditsky, p.6).
- It is likely that the SSO product you choose uses a directory server as a repository. If you already have a directory server, you may need to extend the schema. Otherwise, you may need to introduce a directory server in your infrastructure. You may need to import user data from your existing data repositories into the directory server for you to be able to have SSO. Alternately some of the vendors support RDBMS databases to be the data store for the SSO application. Carefully review the advantages and disadvantages of RDBMS over a directory server and then only choose to use one of them. Going through the best practices document of the vendor of SSO product can help you in this.
- You may need to establish an account with a third party digital certificate authority like Verisign, if you choose to implement communication between different SSO components over a digital certificate. Or you may choose to implement your own certificate-issuing server software such as Netscape or Microsoft Certificate Server.

3.3 Implementation

3.3.1 Determining the architecture

One of the most important aspects of implementation is designing the architecture for the proposed implementation. Architecture forms the foundation of the new system and hence it needs to be robust, flexible, secure and scalable. A thorough understanding of the needed components of the system and where they can reside in the architecture, experience with the design of the architecture and participation of resources from all relevant departments in the architecture design discussions can contribute to a good design of architecture. The rest of this section aims to give the readers a platform for clarity of thought before, during and after architecture design.

A typical architecture in different types of SSO with security in mind is given below. This should only be considered as a guideline and you are expected to discuss this thoroughly with your IT security team before consolidating the architecture. Please note that a generalized and common architecture is given and vendor-specific architectures may vary depending on the components of the product and the way the components interact among themselves. An explanation of the terminology that you find in the following discussion is out of scope of this paper. The references section at the end of this paper lists a good book (reference #12) for beginners to network security that can be helpful in understanding the terminology.

It is important to remember that security of communications between components in architecture plays a key role in the overall architecture. All communications should be encrypted and transmitted across an SSL connection. This includes the communications between the engines (access and identity) and the data repositories.

> The encryption method should rely on certificates, not passwords determined by the installer or shared secrets. This is because somewhere during the implementation there will be a set of "commonly" known passwords between the implementation and support group. The longer the components are in place, the more likely other support personnel will have access to these passwords (Hobbs, p.10).

Before we take a look at the architectures, let us see what the building blocks of enterprise SSO are. They may be called with different names in different products but by and large they share common characteristics.

Access component: As the name suggests, the access component plays the role of a gatekeeper for access to all protected resources. For example, if you are trying to access a URL on a web site that is a part of SSO then the access component intercepts your request, verifies that you are authenticated and authorized to access the URL and then only lets you access the URL. Access component can be further divided into an access agent and an access engine. Access agent resides on the web server. When a user requests to access a protected resource then the access agent verifies if the user is authenticated and authorized to access the resource by contacting the access engine which in turn contacts the data repository to get this information. After authentication, a cookie is stored in the user's web browser or user's hard drive (depending on the type of cookie) and is passed to all the subsequent applications of the same domain that the user browses using the same browser. If the application is a part of SSO, the cookie provides the needed authentication and authorization information and the user is not needed to provide credentials again. The orchestration of SSO in a multi-domain SSO is a little different from this, as explained later in this section.

Identity Component: Identity component provides identity management. If you change the information related to your identity (either as a user or as a group) like changing your profile information or changing your password or changing your group attributes etc. on a web site that is a part of SSO then it is the identity component that works behind the scene to facilitate the functionality. We go with a brief introduction to identity management and then move on to the explanation of constituents of the identity component. The purpose of identity management is to manage the identity of users and groups. A user possesses a few attributes; some of them static like username, last name, first name (assuming that it is rather infrequent for a user to change his name or username), user creation date, etc and some of them are dynamic like password, the department they work for, the designation, date of last password change, etc. Similarly a group can also possess static and dynamic attributes. A group can be categorized into static or dynamic group. An administrator maintains a static group. For example, if Joe Smith joins in IT department of an organization and if the IT group is maintained as a static group then the administrator creates him as a member in the IT static group. If Joe Smith changes to the HR department after some time, he needs to be manually changed in the group configuration to the HR static group. A dynamic group is a group based on a user attribute. For example, there can be a user attribute to store the user's department and depending on the value on this field the user will be a part of the department-related dynamic group. If Joe Smith has a value of IT for this attribute then he belongs to the IT group. If the value is changed to HR then the user automatically belongs to the HR group based on the value of the attribute. Dynamic groups can reduce the maintenance overheads and can lead to better security in identity management. When Joe Smith moves from the IT department to HR then he needs immediate access to HR-related information and doesn't need access to IT-related information and this can happen rather transparently in a group-based identity management.

An identity component may be a part of the SSO product (or offered as an additional component in a few products) that helps in identity management. It is often implemented as an identity agent that resides on a web server and an identity engine. When an authorized user accesses or modifies identity information then the identity agent on the web server communicates with the identity engine, which in turn communicates with the data repository to get the job done. The identity component deals with enforcing restrictions on password management based on policies. A majority of the products come with flexible ways of managing workflow in different situations. In the context of identity component, workflow can be defined as the path that a request to modify user/group identity information is routed through, based on certain criteria. For example, enforcing approvals when certain key attributes like compensation, department, etc. on user profiles are changed.

Administration Component: This comes in different flavors the most common of which is a web-based administration console. It allows an administrator to configure access component (authentication and authorization information like

protected resources, who can access what, etc.) and/or identity component (configuring users, user attributes, groups, etc.).

Monitoring and reporting component: This is an add-on component in most of the SSO products. Generally this component is not given enough focus and hence not properly evaluated. In fact, a lot of implementations are performed without this component in place. However, the component helps to ensure continued security of the implementation by identifying events of special interest (like security policy violations etc.) and notifying the needed people about them. Based on the need, third party products can be integrated with SSO products for providing advanced monitoring and reporting functionality. Again, careful integration is recommended here not to open any vulnerability. It is also recommended to go with a very basic monitoring and reporting functionality in the initial implementation and increase the intensity in subsequent releases.

Data repository: Most often this is an LDAP-compliant directory server that serves as the repository for user data and the metadata. A few vendors support an RDBMS database to be the back end. Some of the database vendors provide an LDAP wrapper to the database as well. In a distributed enterprise deployment that spans multiple geographic locations, it is common to have one or more master servers with synchronization between those servers at regular intervals and a few other servers that replicate the content from the master servers. Strong authentication between the servers before data replication can help increase the security in the distributed architecture.

Now that we have a basic understanding of the building blocks of an enterprise SSO, let us move on to a technical discussion on architecture. As mentioned in the overview of SSO, the presentation of cookie lets a user to navigate into an application without further authentication. Hence it is very important to understand what information is stored in the cookie is and how tough it is to read the contents of a cookie with an attack and then spoof a user cookie by an intruder.

Architecture for Internet facing SSO

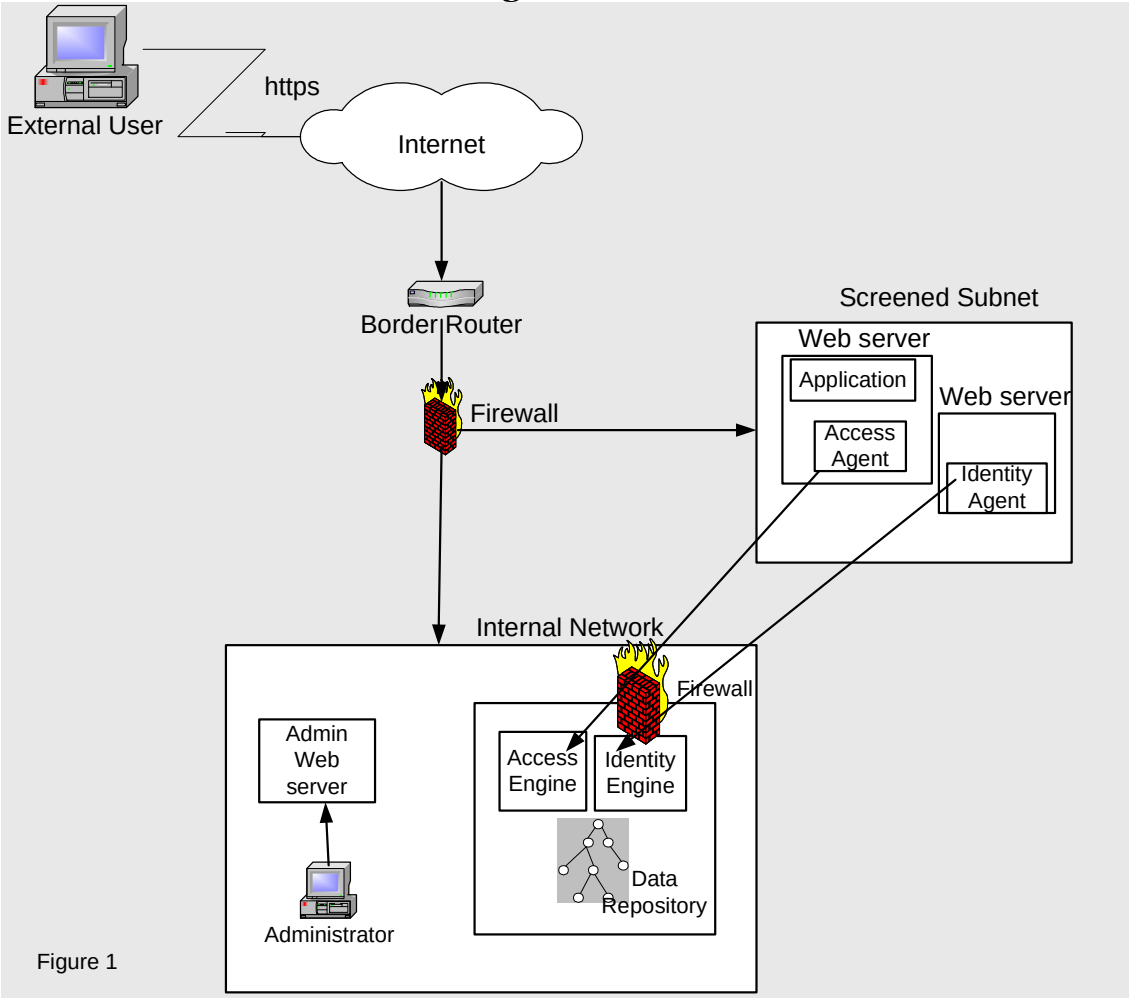


Figure 1

Figure 1 attempts to visualize the location of different components in a common architecture for Enterprise SSO. Internet facing applications are prone to a variety of attacks and hence the concept of defense in depth should be used as a guiding principle in the architecture to ensure maximum security. The web servers that hold access agents and identity agents can be placed in a screened sub-net to separate them from the rest of the network. The access engine, administration server and data repository can all reside in the internal network. The communication between a user and web server can be secured by using Secure Socket Layer (SSL). This avoids sensitive login information (like login and password) getting transmitted in clear text.

Many vendors offer different types of communication between the components on the web server and the components in the internal network. These types can be broadly categorized as clear text and certificate, though vendors may call them differently. In the clear text type, the web server communicates to the application server (access engine or identity engine) without any encryption. It is probably a good choice for a development environment to avoid the hassles of

procuring and maintaining a digital certificate (explained later). Obviously this is very insecure for a production environment and hence the need for the second type, certificate. A digital certificate is an electronic document that identifies an entity like an individual, company, client, server or some other entity. You can visualize a digital certificate to be the electronic driver's license with the Certificate Authority (CA) being the party that endorses it by validating the identity of the entities. They can be either independent third parties like Verisign or organizations running their own certificate-issuing server software such as Netscape or Microsoft Certificate Server. Communication that uses a digital certificate is expected to be secure as long as the keys associated with the certificates are not compromised. The references section in this paper provides a useful link (reference #13) for more understanding on digital certificates. You can place the application servers behind a firewall, allow only the hosts holding SSO components to communicate to the servers at designated ports and deny any other traffic. The firewall can be configured to allow outgoing traffic with source address belonging to your network, only to the hosts holding SSO components and deny all other traffic. This reduces the chance of malicious programs like Trojans sending spoofed traffic from application servers to the Internet. It may take significant efforts to move the data repository if you choose to use an existing one. Communication between access or identity engine and the data repository can be enforced on a secure channel over SSL. It is important to understand at this stage if a vendor supports these security measures or not, if it was not already done during evaluation.

Other layers of defense on the server can be hardening these servers, applying security patches as soon as possible after they are released, restricting access of the servers to only a very limited number of people, designing a corporate workflow when a new resource needs access to any of these servers, logging all important activities on the servers, etc. As a simple defense on the client side, you can define supported versions of different browsers and prevent browsers of older versions to access your applications. You can show a friendly error message to indicate the supported browser versions in such a case. This can help you prevent an attacker taking advantage of a known vulnerability in an older version of the browser used by the user.

Architecture for Intranet facing SSO

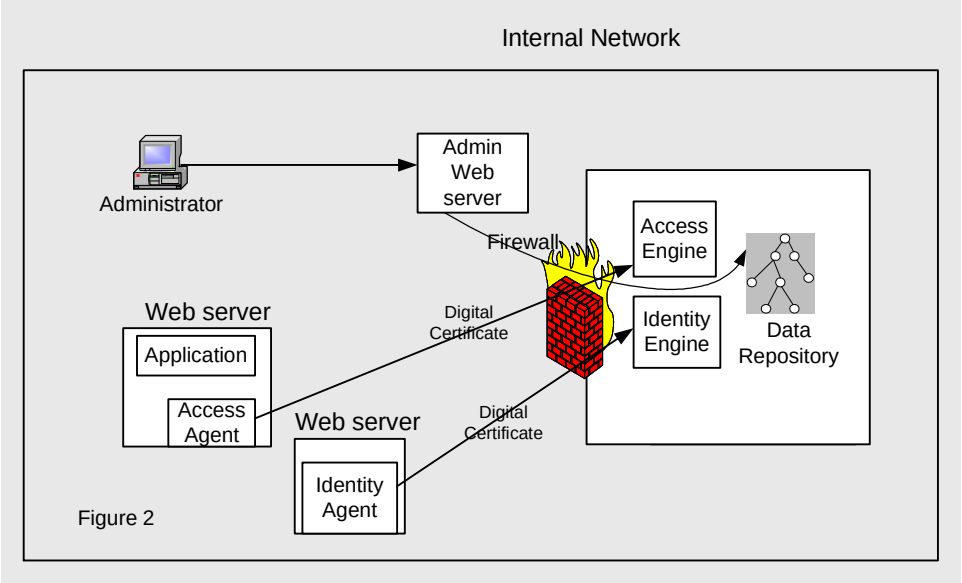
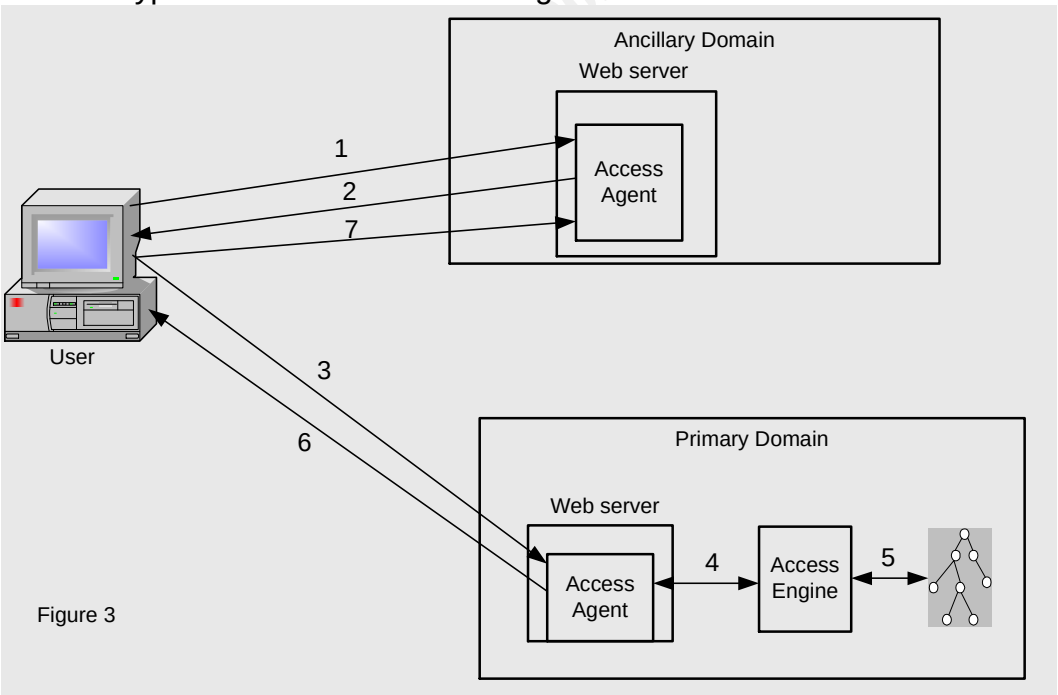


Figure 2 shows the common architecture for Intranet facing SSO. It is similar to that of Internet facing SSO. Since all the components of this architecture reside in the internal network, all the components are placed in one zone - internal network. The other security measures that were discussed for the architecture of Internet facing SSO could still be implemented as the next layers for defense in depth.

Architecture for Multi-domain SSO

As stated before, this can be in two flavors i.e. Intra-organization multi-domain SSO and cross-organization SSO. Traditionally, a similar architecture is used for both the types and it is shown in the figure 3.



For simplicity sake, access control devices such as border routers, firewalls, etc are not shown in the figure. In this type of architecture, there can be one primary domain and one or more ancillary domains. A trust relation is to be established between the web servers in primary and ancillary domains either using a third party trust management tool or by using cryptography using keys known to the access agents. The way of establishing trust can vary between different products and you need to consult the product documentation or discuss with a product specialist to understand how the trust can be established. The access agents in ancillary domains don't have the ability to authenticate the user themselves. When a user tries to access a resource protected by ancillary access agents, the sequence of events is as given below.

1. User uses his browser to access a resource protected by an access agent in the ancillary domain.
2. The access agent sends a response to the user's browser to redirect to the access agent on a web server in primary domain to get authenticated.
3. User's request gets redirected to the web server in the primary domain. After collecting credentials from the user, the request is sent to the web server.
4. Access agent on the web server in primary domain contacts the access engine for authentication and authorization.
5. Access engine gets the needed information by contacting the data repository.
6. Access agent sends the authentication and authorization information back to the user's browser, after it receives the information from the access engine. A cookie is set on the user's browser for the primary domain.
7. The user's browser is redirected to the protected resource in the ancillary domain by adding the encrypted authentication and authorization information in the query string or in a form given by the trust management tool. If the user is authorized to see the protected resource then he is redirected to the requested resource. The ancillary web server then sets a cookie on user's browser for the ancillary domain.

When a user clicks on the logout button on a page in the ancillary domain then the web server in the ancillary domain destroys the cookie set for the user and redirects the user to the primary domain with information to destroy its cookie. The web server in the primary domain then destroys its cookie for the user and the user is logged out of the application.

Note that the behavior can vary drastically in different browsers in this context. This is where functional evaluation discussed above can be handy in analyzing if all the claimed functionality by the vendor can be achieved without any security holes.

This architecture fits well for an intra-organization multi-domain SSO. However, there is a disadvantage if you choose to use this architecture for a cross-organization SSO, as each web server in the partner site should have the access agent installed and configured to be able to talk to the access agent in the primary enterprise domain. These access agents need to be maintained and

upgraded along with any upgrades of the access agents in the primary domain. There is an ongoing attempt to address this issue with the help of federated identity management as described below.

Federated identity management, used in cross-organization SSO, needs a slightly modified architecture as depicted in figure 4.

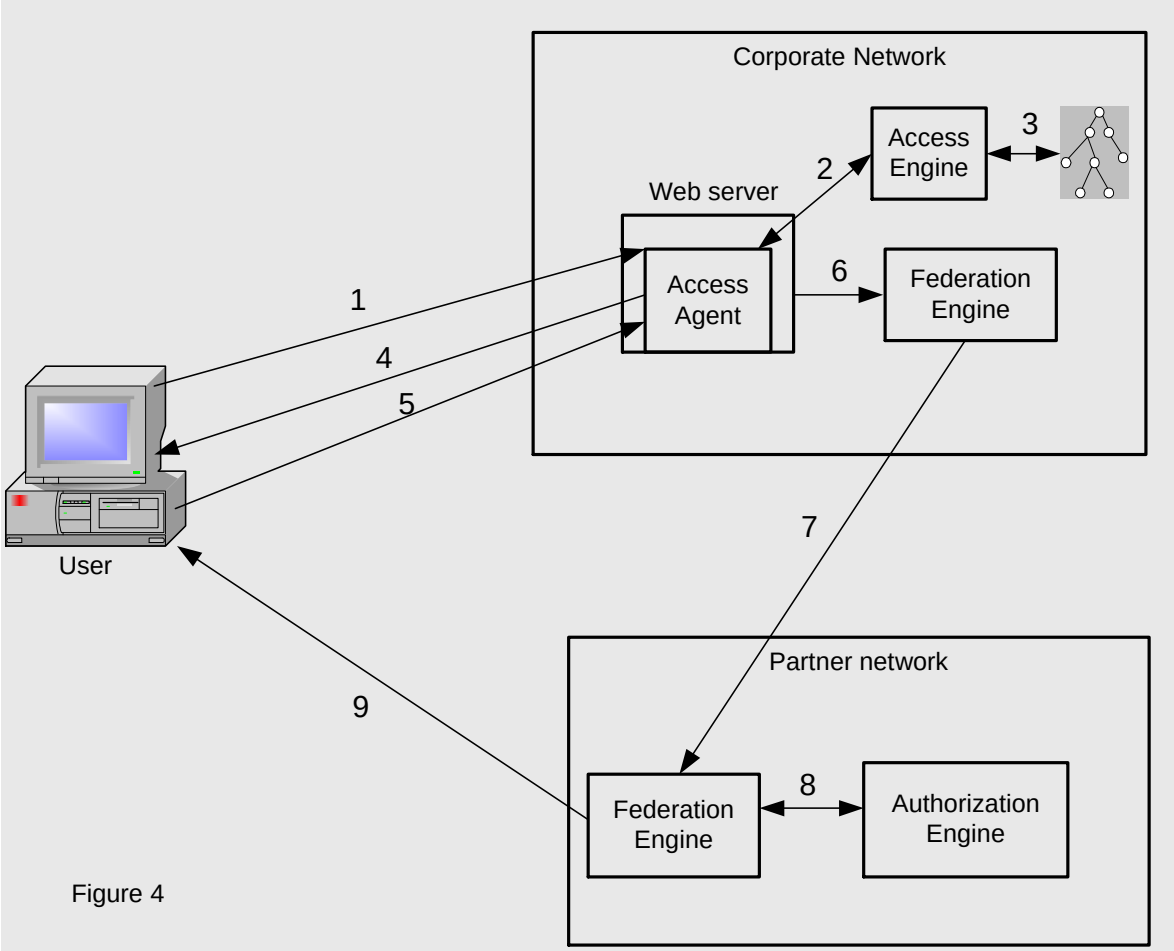


Figure 4

Access control devices such as border routers, firewalls, etc. are not shown in the figure to make it simple for understanding.

1. A user tries to access an application on the web site of an enterprise by providing his credentials.
2. Access agent on the web server contacts access engine for authentication and authorization of the user.
3. Access engine gets the needed information by contacting the data repository.
4. Assuming that the user is authenticated and authorized to access the application then a cookie set on the user's browser.
5. The user then clicks on a link that takes the user to an application maintained on a partner site (Say, an online survey application to gather information on what the user's experience is with the web site).
6. The access agent on the web server of the corporate web site passes the buck to federation engine in the corporate architecture.

7. The federation engine uses the user's browser cookie to pack it as a SAML (Security Assertion Markup Language) assertion, digitally sign it and then redirect it to the federation engine in the partner's network.
8. The federation engine in the partner network receives the SAML assertion, identifies that it was sent by the federation engine in the corporate network because of the digital signature and extracts the user's identity from the SAML assertion. It then passes the identity information to the partner's authorization engine to know whether to allow the user to see the requested application or not based on the authorization of the user.
9. If the user is authorized, the federation engine on the partner's network will redirect the user to the requested application.

This architecture has two inherent advantages.

- The partner doesn't need to maintain the identity information of the user.
- No separate installation is needed on the partner web servers.

Of course, the federated engines on both sites should be configured to identify, trust and interact with each other as designed. But, it gets easier when they are based on standards such as SAML or Liberty. The references section gives resources to give more information on these standards.

The following questions aim to help you in understanding different aspects of security in the architecture of enterprise SSO.

1. What are the different zones in which the components of the SSO product will be located? What are the recommendations from the vendor in this context?

2. How do these components affect security? For example, if there are components residing in DMZ and need to contact servers residing behind a firewall, what type of secure communications does the product support? Which of the options given by the vendor is secure to implement? How does it affect the performance of the overall applications participating in SSO?

3. What is the corporate policy on redundancy at different points of application in the architecture and does the SSO product support design for redundancy? The goal is to not have a single point of failure i.e. if one of the components fails then will it bring down the application or any important functionality of the application like authentication or authorization? For example, what happens if one of the web servers of a web application that participates in SSO goes down? What happens if there is a hardware/software breakdown on the firewall? What happens if the access engine or identity engine has a hardware/OS crash? What happens if one of these servers is disconnected from the network? What is the failover configuration in the architecture? Are there any known security issues in this type of configuration? Does it match the recommended configuration by the SSO product vendor?

4. How is load balancing designed in the architecture? Does the SSO product vendor support load balancing access/identity engines? Does the SSO product use any third party products for this load balancing? If so, are there any known security issues in the version of the product that you may install in your architecture? If Network Address Translation (NAT) is used, will it affect the SSO product functionality?

5. Is disaster recovery built into the architecture?

6. Are resources from information security participating in design discussions for the architecture? Do they see any potential security risks in the architecture?

7. What is the protocol used between the components? Is it proprietary? If so, what is the range of ports that we need to open? Does it open new avenues for attacks by intruders?

8. Does the architecture need any special components to support emailing requirements of the application?

3.3.2 Nailing down requirements

While the discussions about architecture are in progress, it is a good idea to start gathering the functional requirements of the implementation. As mentioned before, it helps to start simple with the initial implementation and take incremental steps towards increased functionality in subsequent releases. Business analysts often face a dichotomy between accepting the available functionality in the product and requirements that need customization. Generally it gets tougher to convince business analysts to accept the available functionality as it implies that they need to knit the business processes around the functionality. However, the recommended practice is to minimize any customization. The vendor of a product can invest significant efforts in making the product more secure. You may not be able to secure the customizations developed in-house to the extent that the vendor can ensure in his product line. If you have already developed a prototype as described in the evaluation section, then by this time you have some in-house resources who understand which of the requirements can be implemented through configuration and which of those need customization. So, prioritize your requirements, interact with the available technical resources and reprioritize those requirements with minimizing customization as the goal. You must cautiously balance security and usability in your requirements and look for solutions that provide both. The following gives a few questions that can help you nail down your requirements.

1. What are the applications that I can choose to participate in SSO in the first implementation? These are often the applications that are not directly related to revenue generation of an organization. The goal is to try the first implementation on applications that affect the organization to the least if they

are taken down for longer than anticipated time during the production migration or if they are taken down frequently because of unforeseen functional or security issues observed after migration to the production environment.

2. What resources in these applications need to be protected? For example, which URLs need authentication and authorization before they can be accessed?

3. What restrictions are to be enforced on passwords to these applications? Did the members of security team review them?

4. Are there any emailing requirements? For example, if there is a requirement to expire a user password, does the user need to be informed through email in a specified number of days before expiring the user password?

5. What is the forgotten password mechanism? A majority of the applications use a security question and answer as a way of self-service to recover password. Is this forgotten password mechanism approved by IT security team members? If so, did they add any new requirements to ensure that password recovery does not introduce new security holes?

6. Are there any other requirements related to password security? For example, do you need to enforce a user to change the password on the first login to adhere to your corporate policy?

7. Are there any known conflicts between the corporate security policy and product functionality? What is the planned course of action in such situations?

8. What are the auditing, monitoring and reporting requirements?

9. What are the application log requirements?

10. What are the workflow requirements?

3.3.3 Investing right resources

Involving the right resources from the early stages of the project is key to the success of implementation. Obviously, these resources generally have a very busy work schedule and hence it may be difficult to pull them into the project meetings. Maintaining systematic documentation on the discussions and developments in each project meeting is a task of paramount importance. The documentation should be available to these resources for reference (on an intranet website or on a shared drive). Following are a few questions to help a project manager in resource management.

1. Did you identify the needed resources for the successful initial SSO implementation and schedule their time for the project? Internal resources like developers, project managers, security professionals, business representatives, business analysts, network administrators, firewall administrators, directory server administrators, quality assurance engineers, application specialists, trainers, help desk agents, etc. and external resources like points of contact on the vendor side, contractors, etc. need to be identified. Review the needed resources after evaluation of the products and try to fill in the gaps, if any. Discussion with the SSO product vendor, about the needed resources for the project, based on their experience in prior implementations also can be of a great help in this aspect.

2. Do you have the needed in-house expertise for the implementation of the project? Often it helps to engage a team of experienced consultants to lead the implementation while the in-house resources work with them and learn over a period of time. Did you visit the need of engaging such a team of consultants?

3. Did you establish communication channels for these resources to interact with each other during the project? They can be intranet websites for posting project documentation, mailing lists and discussion groups for a common place to post questions, etc.

4. Did you plan for the resources that will be primarily responsible for documenting the progress in project meetings?

5. Did you identify and plan the needed time for training the needed resources? The training can be in different forms like end user training, train the trainer, technical and security training for the development and security and project management teams from the product vendor, etc.

6. Did you plan for the time of the resources during production migration? There can be long migration times for SSO implementations based on the applications that are integrated.

7. Did you plan for the lifecycle of a help desk request about issues in SSO or password management after production migration, right from the point when a help desk agent receives a call or email or some other form of request to the time when the issue is resolved and communicated back to the user?

8. Did you document agreed upon Service Level Agreement (SLA) with the vendor for raising issues/creating service tickets after implementation, definition of severity levels, escalation procedures, communication channels, etc.

3.4 Common mistakes

3.4.1 Process/Project mistakes

1. SSO is one of the requirements of a project: Implementation of SSO is major project unto itself. Many a time it is observed that SSO is just one of the requirements of a project and to satisfy the requirement, the implementation is taken up resulting in impractical schedules in the project plan. It is a recommended practice to take time to discuss with another customer who implemented an SSO product of your choice to understand the usually faced challenges and plan accordingly for the project.
2. Too ambitious about SSO implementation: Trying to take a big leap in the first step can be a good recipe for disaster. As mentioned in the planning section, divide the whole SSO project into small and manageable units and take incremental steps towards the full-fledged implementation.

3. Project manager's experience is the only important thing: Project manager is going to orchestrate resources from different departments. An experienced and security savvy project manager with the skill set and ability to understand and verify every step of the project with security in mind can take a lion's share in the success of the project. If you don't have a project manager with the needed security awareness then, it helps to train the available project manager on the essential security aspects of such a project. Ignoring the needed security awareness, skill set or training for the project manager is a big mistake that has the potential to lead to a catastrophe.

4. No cushion time allotted in the project plan: Expect the unexpected. Not giving enough cushion time to absorb unexpected delays in the first implementation of SSO can lead to confusion and chaos when unanticipated challenges come your way.

5. Communication about the SSO system within the organization is not planned: People in the organization, other than those involved with the project, often come to know about the new SSO system just before the release of the project. They are initially overwhelmed about what they are going to get and then get frustrated with the new password restrictions and other security enforcements. Starting the communication with user community from the early stages of the project can help set the expectations of potential users. You may choose to post this information about the new SSO system on your intranet sites, bulletin boards, etc. or use email communication at regular points of time.

6. Quick evaluation to save time: Many a time not enough time is spared for a systematic evaluation. It is done based on the feature comparison or vendor preference or referral etc. Since each environment is unique, it is worth spending time on planned evaluation, as it will result in saving of a lot of

efforts, cost and frustration in the long term. Consideration of long-term requirements may be a step in the right direction. For example, if you see a need for implementing legacy SSO in the near or distant future, evaluation with a goal of how well the SSO product can deal with legacy SSO is probably appropriate.

7. Resource from IT security is not engaged from the beginning of the project: It is highly recommended that an experienced resource from IT security be a part of the project, from the inception to at least the end of initial implementation. Identification of any potential security issues early in the game can alleviate a lot of stress and effort in the end. Also, the resource from IT security can try to educate the project manager and the other resources on the project, thus increasing the security awareness in the team that can have a positive tangible effect on the success of the project.

8. Consultants leave the project with insufficient documentation or inadequate knowledge transfer: It is common to hire consultants in the initial implementation to help in project development and guidance, based on their experience. Often, because of ambitious timelines and shrunk budgets, the consultants leave in a short notice. So, they will have no time to share their knowledge either in the form of detailed documentation that could be a good reference in future or in the form of dedicated sessions for transferring the needed knowledge to the in-house teammates. They quickly explain the "most important aspects" of the implementation and move out of the project. Ideally, there should be a gradual transfer of control from consultants to the in-house resources. If possible, during this transition period, the consultants should shadow when the in-house resources manage the show. This avoids the in-mates getting overwhelmed if something goes wrong as soon as the consultants leave.

9. Creating a requirement because a feature is available in the SSO product: Based on a demo or evaluation of the SSO product, it is easy to get tempted to insert a new requirement because there is a nice feature available in the SSO product that you just selected to implement. However, the driver for requirements should be a business need, not an available feature in the SSO product. Switching on new functionality may introduce new security, compatibility or other issues. It is recommended that you come up with a bare minimum set of requirements that is mandatory for the initial implementation after discussions with the project resources and stick to it till its successful completion. All additional requirements can be labeled as enhancement requests and can be taken up in subsequent maintenance.

3.4.2 Technical mistakes

1. Choosing between certificates and shared secret without a thorough understanding: Vendors claim that they support encrypted communication

between different components of the SSO product, giving a choice between different modes of communication. You need to understand all the dimensions of implementing different forms of communication before choosing one. For example, if there is a common shared secret used by all the components of the product then the whole infrastructure that is based on SSO can be compromised by an attacker if he knows the shared secret. So, usage of different shared secrets for different components is better. However, an attacker could manage to know all the shared secrets if they are used over prolonged period of time. Hence, there should be a way to "renew" these shared secrets. It is a concern of single point of compromise even if the communication uses digital certificates and just one certificate is used for the communication between all the SSO components. On the other hand, if you use different third party digital certificates for communication between different components then the cost of procuring and maintaining those certificates can be high. More over, these certificates expire after a specified amount of time, usually one year. So, if you forget to renew and install the renewed certificates on the appropriate servers then it may bring the entire infrastructure based on SSO to a screeching halt. There could be many more aspects like this that you may need to understand before getting a feel of comfort with any option.

2. Under-estimating issues around cookies: Since HTTP is a stateless protocol, usage of cookies is the most widely used mechanism in web-based applications to maintain the state of a user throughout a session. However, there have been a variety of attacks based on cookies and hence it is important to understand different aspects of cookie management. The very fact that a cookie is encrypted does not mean that it is secure. Focus on details like the information stored in a cookie (if there is any user identity information like user id, password, session id etc.), the algorithm used to encrypt the contents, whether the cookie is a per session cookie or persistent cookie (explained in overview), how well it works if a client IP address is stored in the cookie and a firewall or a load balancer modifies the client IP address while proxying connections, how easy or tough is it to launch a session hijacking or replay attacks after noting the cookie contents, etc. can take you a long way in getting the big picture of cookie management in the SSO application.

3. Bumping up to the maximum logging option without understanding what it means: Many a time it is observed that the technical team or the decision makers of the project like to play it safe in the initial implementation by bumping the logging option to the maximum on all the components. The idea behind this is to get as much details in the log files as possible in the early stages of initial implementation for easy troubleshooting of any issues. While the intention is good, it suffers from the following disadvantages.

- It can quickly fill the disk space, leading to an unpredictable behavior of the underlying operating system. It adds an overhead on the administrators to keep monitoring the log size and archive them.
- Depending on the information the SSO product writes into the log files in its maximum log option, it could be help a malicious user to get sensitive information. For example, if the SSO product writes user credentials into the log file in the maximum log option then it could prove to be an attacker's paradise.

Hence, it should be an informed decision to increase the log option. Measures like discussions with resources on the vendor team, consulting SSO product documentation, etc. can help greatly in this direction.

4. Ignoring potential issues with failover configuration: Probably all of the SSO vendors claim that they have software failover as a feature built into their product. It usually means that you can maintain redundant software components in the architecture and configure them to manage the show when the primary software components fail. However, it needs a closer look to understand what it can get translated into in terms of failover. The following are a few questions that can help you in this aspect. Some of the terms used below are explained in the architecture section.
   - What is the mechanism by which the access/identity agent realizes that a corresponding access/identity engine has failed?
   - How quickly can access/identity agent respond to such a failure? How quickly can access/identity engine realize the failure of a data repository? Are they configurable parameters? Is there any side effect on modification of this configuration?
   - How does this response time get affected if the communication between these components is going through a proxy firewall? Does it need to hand-in-hand with a special configuration on the firewall?
   - Is the behavior going to be the same even when the network connectivity between access/identity agent and the corresponding access/identity server is broken suddenly?

5. Expectation of plug and play with firewalls: Whatever the documentation from the SSO product vendor or from the firewall vendor may claim, it remains a fact that the introduction of firewall in the architecture is seldom plug and play. You need to expect some direct or indirect challenges, especially if you plan to a proxy firewall. A direct challenge is something that shows you very evident symptoms that it is a communication issue and often it is easier to troubleshoot. For example, if you enter your login and password in the login form and wait indefinitely to access a resource then it is probably a communication issue between the SSO components. An indirect challenge shows an arcane symptom that may seem quite unrelated to communication. For example, your application may show strange behavior sporadically that may point to the issue in communication after a tedious session of

troubleshooting. Because typically you don't use firewalls in a development environment, you might not see these errors during the product evaluation. However, you can use the same fact as your tool to compare the configuration between your development and QA/Production environments to understand the differences and nail down to the communication issues quickly.

6. No understanding on the effects of turning on/off of the features or other aspects of configuration offered by the SSO product: When a product is installed then it may come with a few functions/options switched on/off by default. So, while evaluating the product everything may look like working great but, when you choose to switch on/off one or more functions/options in the development/QA/Production environments for getting rid of complexity or increasing security or for some other reason, it may start cracking the SSO application. So, having an open eye for the effects of switching on/off the features/functions/options in the product is important.

7. Ignoring the need for automated migration of configuration between environments: The capability to migrate configuration from one environment to another is something that you may assume to have in the SSO product. However, there are a few SSO products where this feature is not available. It not only means a lot of effort to manually migrate every piece of configuration between environments but also is error prone because it is manual in nature, thus leading to possible security or other issues.

8. Internal zone doesn't need secure communication: Very often, there is a sense of security that prevails when you think about the internal zone. However, history shows that the number of attacks on the systems in the internal zone is at least equal to those on the external facing systems. Furthermore, traditional security measures like firewalls, intrusion detection systems etc. cannot protect against unauthorized insider access. So, securing the communication of sensitive information in the internal zone is of high importance and cannot be neglected.

4. Post-Implementation verification

4.1 Testing the application
Often, a majority of the efforts are spent in functional testing. Quality assurance resources should be in consistent interaction with development, security and integration teams to make sure that they succeed in testing the application completely, in all aspects.

4.1.1 Functional testing
This includes tests to verify that SSO works properly and other requirements like restrictions on password, enforced change of password, lost password retrieval, password expiration, workflow requirements, protection of resources, etc. are met. Use different browsers (like Netscape, IE, Mozilla, Opera, AOL, etc) and different versions of each browser based on the supported browsers and their versions of each application to perform this testing.

4.1.2 Integration testing
SSO application integration is a very fragile area and hence it needs to be systematic. The purpose of integration testing is to ensure that each application that participates in SSO works the same before and after SSO. So, thorough regression testing of each application of SSO is essential. Using different browsers and their versions should be done in integration testing to ensure that using every supported version of a browser results in successful navigation to all parts of the integrated applications. Automated testing with the help of tools can help increase the speed of testing.

4.1.3 Security testing
Ensuring that the implementation of the SSO product doesn't leave any security holes is a very challenging task. Hence, it is important to test all existing applications and look for security holes or weaknesses. Training the QA resources in the needed skills for testing the security of an application can be of a great help. Following gives a few tips to help you in this task.

- Follow the mailing lists and discussion groups for the SSO product and its peers.
- Follow the mailing lists and discussion groups of the vendors of any existing application (that is a part of SSO) to find out any security holes in these applications that may be a result of the integration.
- Preparing a methodical test plan for testing possible security holes or weaknesses, after discussions with in-house security team.

The following are some other steps that can help you in the same direction.

Vulnerability scans: Vulnerability scanning can help you to secure your network by identifying and fixing any weaknesses before an intruder can realize these weaknesses. Nessus is a popular, freely available and open-source vulnerability scanner while there are a variety of commercial tools available in the market.

Buffer overflow testing: Buffer overflow occurs when a program tries to write more data into memory than the space allocated, resulting in unexpected behavior. It can be a result of poor programming practices and a lot of exploits have so far been based on buffer overflow. One of the ways of testing buffer overflow in applications is to populate the fields in the user interface in different parts of the application to the maximum and test to see if it breaks the application. For more information and quick understanding on buffer overflow and countermeasures, refer to #18 in the references section at the end of this paper.

SQL Injection testing:

> The principle of basic SQL injection is to take advantage of insecure code on a system connected to the internet in order to pass commands directly to a database and to then take advantage of a poorly secured system to leverage an attacker's access. Interestingly, this is also one of the results of poor programming practices and developers who use 'string-building' techniques in order to execute SQL code usually cause it (McDonald, p.3).

The references section at the end of this paper lists a resource for detailed understanding of SQL injection attacks. Try to test the user interface of the applications in SSO (including the administration console and all other user interface of the SSO application itself) with a variety of non-alphanumeric characters like semi colons, commas, single quotes, parenthesis, etc. Try to input some complete and incomplete SQL statements in the fields to see if it breaks the application.

Malformed URL and other DOS attack tests: A Denial of Service (DOS) attack can be defined as an explicit attempt by a malicious user to prevent the usage of a system by authorized users. It can be better explained with an example for the benefit of the readers. I can launch a Denial of Service attack on a pizza restaurant by ordering hundred (or more) pizzas from the pizza restaurant by spoofing the names and phone numbers of all my neighbors. Then a genuine person who wants order a pizza from that restaurant will not be able to do so because the staff at the pizza restaurant is overwhelmed by the orders I gave. DOS attack can be categorized into different subcategories based on the way of attack. One of the common ways of launching a DOS attack against a web site is to periodically send a URL constructed with non-alphanumeric and irregular characters including Unicode characters. Quality assurance resources can interact with security team to understand the different ways of launching a DOS attack and test it against the applications that are part of SSO, including any web-based administration consoles. A resource to provide a comprehensive understanding on ways of minimizing, detecting and reacting to DOS attacks is provided at the end of this paper.

Backdoor attack tests: Applications can leave a few back doors that can be used to bypass layers of defense. It is important for Quality Assurance resources to be aware of common types of back doors in the context of SSO to see if a user can bypass restrictions imposed by the SSO architecture. The following gives a

few examples to throw some light on these aspects. Please note that this list is no way complete. The users can go through a lot of resources on the Internet (like discussion groups, mailing lists etc.) and use their creativity to find out many such possible ways of bypassing the protection of the resources.

- Can a user disable JavaScript in his browser and bypass validation in the user interface?
- What is the expected behavior if the user disables cookies in his browser?
- Can a user modify the hosts file on his workstation to give a fictitious name to the hostname of the protected resources and bypass authentication? For example, say the protected resource is http://anyhost.samplecompany.com/protectedresource. I can modify the hosts file on my workstation (in the windows operating system it is typically in the C:\WinNT\System32\drivers\etc folder) to add a new entry with the IP address of host of the protected resource and www.mysneakyhost.com as the hostname. In this case, can I bypass the authentication and authorization using http://www.mysneakyhost.com/protectedresource? Is a similar attack possible through a script?
- Can a user bypass the authentication using an IP address in the URL instead of using the hostname?
- Can a user just refresh the browser or the application to bypass the session timeout?
- If there is a HTML page for entering credential information, is it using https or http? It should be using https to protect against sniffing.
- Can a user enable an option in his browser to warn the user when there is a cookie, watch the contents of a cookie, manipulate them in a script, post it to the web server and escalate his privileges? Can a user use a similar technique to view the profile or any other sensitive information of other users?

4.1.4 Failover testing

A failover test consists of intentionally bringing down one of the components in the architecture of an application to understand the impact it makes on the application. A good architecture design implements redundancy, clustering or failover and load balancing at different points in the architecture so that there is no single point of failure for the application. The failover test plan can consist of

- Shutting down the needed process on different components. For example, for a web site with windows web server, it is done by bringing down IIS Server Service. On a Unix system, it can be done by executing a command to shut down the web server process.
- Switching the power off for different hardware components in the architecture
- Simulating hard drive crash
- Pulling off the network cable from one of the hosts in the architecture

5. Ongoing maintenance

Secure architecture of a system is a journey, not a destination. As time passes, new vulnerabilities will be disinterred in every application. Following is a quick list

of some of the important measures to have in place to ensure security in the architecture.

- Deploy all relevant security fixes and software updates for all software components in the architecture.
- Perform vulnerability scans regularly, on a periodic basis, preferably every month.
- Interact with your IT security team and let them schedule security audits of the system periodically.

6. Conclusion

A well-planned and carefully deployed Single Sign-on product can be a great complement to the other security measures that are already in place in an organization. By weighing the risk factors associated with implementing each SSO product against the advantages and by keeping the expectations aligned with realistic planning, an SSO product implementation to satisfy your requirements is achievable.

The reference section lists a few eminent products in the industry.

7. References

1. Smith, Mick. “Evaluating Enterprise Single Sign-on: A nontechnical guide“ January 2004. URL: http://www.protocom.com/whitepapers/evaluating_sso_nontech.pdf
2. “Enterprise Single Sign-on: Balancing Security & Productivity“. URL: http://www.bnx.com/pdf/Enterprise%20Single%20Sign-On%20-%20Balancing%20Security%20&%20Productivity.pdf
3. Lenox, Greg. “Unraveling the buy VS build conundrum”. 06 March 2003. URL: http://searchcio.techtarget.com/originalContent/0,289142,sid19_gci884437,00
4. Taylor, Laura. “Understanding Single Sign-on”. May 28 2002. URL: http://www.intranetjournal.com/articles/200205/se_05_28_02a.html
5. Sandhu, Sandeep Singh. 30th January 2004. URL: http://www.giac.org/practical/GSEC/Sandeep_Sandhu_GSEC.pdf Version 1.4b
6. Single Sign-On Deployment guide URL: http://developer.netscape.com/docs/manuals/security/SSO/contents.htm
7. The open group. June 1997. X/Open Single Sign-on Service (XSSO) — Pluggable Authentication Modules. URL: http://www.opengroup.org/onlinepubs/008329799/toc.pdf
8. Hobbs, Russell. Considerations For Implementing Single Sign-On Within The Enterprise. 1 September 2003. URL: http://www.sans.org/rr/papers/6/1215.pdf Version 1.4b
9. LDAP FAQ. URL: http://www.mjwilcox.com/ldap/ldapfaq.htm

10. Timothy A. Howes, Mark C.Smith, and Gordon S.Good. Understanding and deploying LDAP Directory services New Riders Publishing, First edition 1999.
11. Boroditsky, Marc and Pleat, Bruce. Security @ The Edge: Making Security and Usability a Reality with SSO. URL: http://www.passlogix.com/media/pdfs/security_at_the_edge.pdf
12. Mailwald, Eric. Network security: a beginner's guide New York: Osborne/McGraw-Hill, c2001
13. Gornes, Ferdinand. SECURITY ALERT: Fraudulent Digital Certificates. 07 June 2001. URL: http://www.sans.org/rr/papers/index.php?id=679
14. Oblix Inc. An overview of federated identity architecture from Oblix URL: http://www.oblix.com/resources/whitepapers/fedid_arch/obx81a_wp_federation_paper.pdf?WP=26
15. Dornan, Andy. Security Assertion Markup Language. 05 December 2003. URL: http://www.networkmagazine.com/shared/article/showArticle.jhtml?articleId=16600124
16. Liberty Alliance Project. A white paper on Liberty protocol and identity theft. February 20 2004. URL: http://www.projectliberty.org/resources/whitepapers/Liberty_Identity_Theft_Whitepaper.pdf
17. Bradley, Tony. Introduction to Vulnerability scanning. 04 March 2004. URL: http://netsecurity.about.com/cs/hackertools/a/aa030404_p.htm
18. Grover, Sandeep. Buffer overflow attacks and their countermeasures. March 10, 2003. URL: http://www.linuxjournal.com/print.php?sid=6701
19. McDonald, Stuart. SQL Injection: Modes of attack, defense, and why it matters. April 8 2002. URL: http://www.sans.org/rr/papers/index.php?id=23 Version 1.4
20. CERT Coordination center. Managing the threat of Denial of Service attacks. October 2001. URL: http://www.cert.org/archive/pdf/Managing_DoS.pdf Version 10.0
21. Lee, Paul S. Cross-site scripting 1 September 2002. URL: http://www-106.ibm.com/developerworks/security/library/s-csscript/

SSO Product links

22. Netegrity Siteminder URL: http://www.netegrity.com
23. Oblix COREid URL: http://www.oblix.com
24. Passlogix v-Go SSO URL: http://www.passlogix.com
25. Protocom SecureLogin URL: http://www.protocom.com
26. RSA Sign-on Manager URL: http://www.rsasecurity.com
27. IBM Tivoli Global Sign-on, Access Manager, Identity Manager and other related solutions URL: http://www.ibm.com
28. Evidian SSO Xpress, Secure Access Manager, Identity manager and other related solutions URL: http://www.evidian.com
29. Computer Associates eTrust access and identity solutions URL: http://www.ca.com

Last Updated: August 16th, 2017

Upcoming SANS App Sec Training

Community SANS Minneapolis DEV534**	**Minneapolis, MN**	**Aug 25, 2017 - Aug 28, 2017**	**Community SANS**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
Secure DevOps Summit & Training	**Denver, CO**	**Oct 10, 2017 - Oct 17, 2017**	**Live Event**
SANS Seattle 2017	**Seattle, WA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS Austin Winter 2017	**Austin, TX**	**Dec 04, 2017 - Dec 09, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017	**Washington, DC**	**Dec 12, 2017 - Dec 19, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017 - DEV522: Defending Web Applications Security Essentials	**Washington, DC**	**Dec 14, 2017 - Dec 19, 2017**	**vLive**
SANS Security East 2018	**New Orleans, LA**	**Jan 08, 2018 - Jan 13, 2018**	**Live Event**
Cloud Security Summit & Training 2018	**San Diego, CA**	**Feb 19, 2018 - Feb 26, 2018**	**Live Event**
SANS San Francisco Spring 2018	**San Francisco, CA**	**Mar 12, 2018 - Mar 17, 2018**	**Live Event**
SANS Northern VA Spring - Tysons 2018	**Tysons, VA**	**Mar 17, 2018 - Mar 24, 2018**	**Live Event**
SANS 2018	**Orlando, FL**	**Apr 03, 2018 - Apr 10, 2018**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**