Single Click Sign on Multiple Websites Using Multiple Credentials

V. Senthilkumar, S. Rudra Ganesh, R. Arunkumar

B. Tech-Information Technology,

CMS College of Engineering, Namakkal, TamilNadu, India-637003.

senthilkumarfreedom@gmail.com, rudra.sparky@gmail.com, blockbusterarun6@gmail.com

C. Yalini,

Assistant Professor,

Department Of Computer Science and Engineering,

Kongunadu College of Engineering and Technology, Musiri, Trichy, TamilNadu, India.

yalini08@gmail.com,

***Abstract*-Single Click Sign On Multiple Websites using multiple credentials is replacing conventional solutions based on multiple, domain-specific credentials by offering an improved user experience. This will allow the user to work with the specified browser only and also it is less securable because of session tracking. Due to these problems this project performs simultaneous logging with the various user credentials on several websites to improve the quality of browsing and reduce the process of log in the websites by the user.**

Keywords-single click SSO; credentials; multiple websites; session; simultaneous logging; user logging.

I. INTRODUCTION

A. WEB SERVICES

A Web service is a method of communications between two electronic devices over the World Wide Web. It is a software function provided at a network address over the web with the service always on as in the concept of utility computing. The W3C defines a Web service as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processing format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. Web services are application components where they communicate using open protocols and are effectively self-contained and self-describing. Web services can be discovered using UDDI, also HTTP and XML is the basis for Web services.

FIGURE 1.1

B. CHARACTERISTICS

Interoperability has Highest Priority. When all major platforms could access the Web using Web browsers, different platforms couldn't interact. For these platforms to work together, Web-applications were developed. Web-applications are simply applications that run on the web. These are built around the Web browser standards and can be used by any browser on any platform. Web Services take Web-applications to the Next Level By using web service your application can publish its function or message to the rest of the world. Web services use XML to code and to decode data, and SOAP to transport it (using open protocols).With Web services, your accounting department's Win 2k server's billing system can connect with your IT supplier's UNIX server.

Web Services have Two Types of Uses:

Reusable application-components:

There are things applications needs very often. So why make these over and over again. Web services can offer application-components like currency conversion, weather reports, or even language translation as services.

Connect existing software:

Web services can help to solve the interoperability problem by giving different applications a way to link their data. With Web services you can exchange data between different applications and different platforms.

II. SYSTEM ANALYSIS

A. EXISTING SYSTEM

The Existing system is browser dependent.There is a possible of cracking of user history from browser based sessions. Its having the separate function to maintain for remembering user credentials when it need for login.It is basis of SSO (Single sign-on).

1. DRAWBACKS

Single point of failure. Single high-value target (attracts more attackers)

2. PROPOSED SYSTEM

It will be an integrated application for all browsers.It could maintain separate database to store user history and credentials for high security. It initially logged in the websites simultaneously which are frequently accessed by the user. It Perform single click sign on with the multiple sign in function to reduce the user load.

3. ADVANTAGES

It is a browser independent system and web integrated system. This will allow the user to work free and safe. The system provides simultaneous logging on the websites with the ease of single click. Categorization of user credentials makes the user to log in the user's favourite websites.

Considering to the existing system the proposed system produces high integrity, safety on credentials and user friendly environment.

B. FEASIBILITY STUDY

SSO capability for Merck's internally developed applications. SSO support for Merck and affiliate organization sites. Centralized access management of Web content for secure administrative efficiency. A flexible solution that will support the current Merck infrastructure architecture and system and will support new requirements. While Merck's current technology supported SSO there were issues as to compatibility, scalability and interoperability. Questions arose regarding the interface with Lotus Domino, tailoring of applications and adaptation of the infrastructure.

C. TECHNICAL FEASIBILITY

Conceptual feasibility study - The study addressed issues such as:

- How can this be accomplished with the current architecture?
- What, if any, changes must be made to achieve the objectives?
- Is there a better product, other than the one currently being used, to support Merck's immediate needs?

III. SOFTWARE ENVIRONMENT

A. DOT NET FRAME WORK

.NET Framework (pronounced *dot net*) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large library and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (as contrasted to hardware environment), known as the Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. The class library and the CLR together constitute .NET Framework.

.NET Framework's Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their own source code with .NET Framework and other libraries. .NET Framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an integrated largely for .NET software called Visual Studio.

1. DESIGN FEATURES

a. INTEROPERABILITY

Because computer systems commonly require interaction between newer and older applications, .NET Framework provides means to access functionality implemented in newer and older programs that execute outside .NET environment. Access to COM components is provided in the System. Runtime Interop Services and System Enterprise Services namespaces of the framework; access to other functionality is achieved using the P/Invoke feature.

b. COMMON LANGUAGE RUNTIME ENGINE

The Common Language Runtime (CLR) serves as the execution engine of .NET Framework. All .NET programs execute under the supervision of the CLR, guaranteeing certain properties and behaviours in the areas of memory management, security, and exception handling.

c. LANGUAGE INDEPENDENCE

.NET Framework introduces a Common Type System (CTS). The CTS specification defines all possible data types and programming constructs supported by the CLR and how they may or may not interact with each

other conforming to the Common Language Infrastructure (CLI) specification. Because of this feature, .NET Framework supports the exchange of types and object instances between libraries and applications written using any conforming .NET language.

d. BASE CLASS LIBRARY

The Base Class Library (BCL), part of the Framework Class Library (FCL), is a library of functionality available to all languages using .NET Framework. The BCL provides classes that encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction, XML document manipulation, and so on. It consists of classes, interfaces of reusable types that integrates with CLR (Common Language Runtime).

e. SIMPLIFIED DEPLOYMENT

.NET Framework includes design features and tools which help manage the installation of computer software to ensure it does not interfere with previously installed software, and it conforms to security requirements.

f. SECURITY

The design addresses some of the vulnerabilities, such as buffer overflows, which have been exploited by malicious software. Additionally, .NET provides a common security model for all applications.

B. VISUAL BASIC .NET AND C#

Visual Basic is a programming language and development environment created by Microsoft. It is an extension of the BASIC programming language that combines BASIC functions and commands with visual controls. Visual Basic provides a graphical user interface GUI that allows the developer to drag and drop objects into the program as well as manually write program code.

Visual Basic, also referred to as "VB," is designed to make software development easy and efficient, while still being powerful enough to create advanced programs. For example, the Visual Basic language is designed to be "human readable," which means the Visual Basic program source code can be understood without requiring lots of comments. The also includes features like "IntelliSense" and "Code Snippets," which automatically generate code for visual objects added by the programmer. Another feature, called "AutoCorrect," can debug the code while the program is running.

C. MySQL

MySQL, pronounced either "My S-Q-L" or "My Sequel," is an open source Relational Database Management system. It is based on the structure query language (SQL), which is used for adding, removing, and modifying information in the database. Standard SQL commands, such as ADD, DROP, INSERT, and UPDATE can be used with MySQL.

MySQL can be used for a variety of applications, but is most commonly found on Web Services. A website that uses MySQL may include Web pages that access information from a database. These pages are often referred to as "dynamic," meaning the content of each page is generated from a database as the page loads. Websites that use dynamic Web pages are often referred to as database-driven websites.

Many database-driven websites that use MySQL also use a Web Scripting language like PHP to access information from the database. MySQL commands can be incorporated into the PHP code, allowing part or all of a Web page to be generated from database information. Because both MySQL and PHP are both open source (meaning they are free to download and use), the PHP/MySQL combination has become a popular choice for database-driven websites

IV. SYSTEM DESIGN

A. DATA FLOW DIAGRAM

A data-flow diagram (DFD) is a graphical representation of the “flow” of data through an information system.

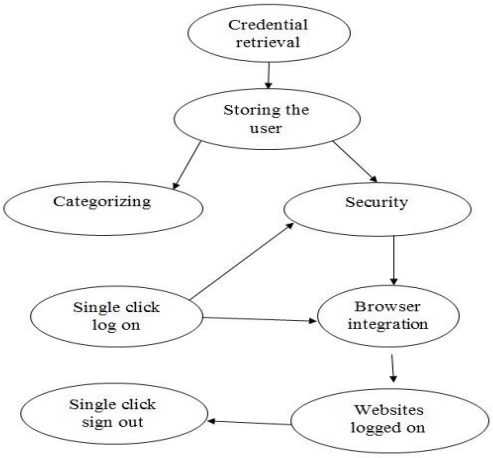
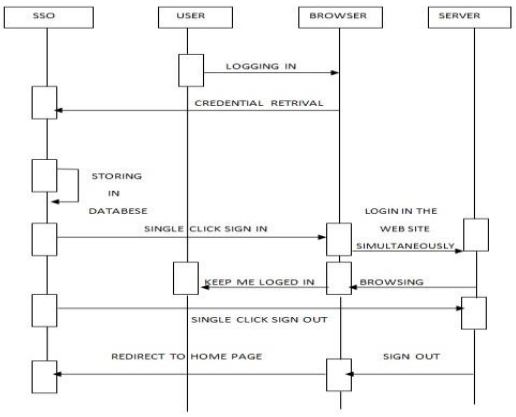


FIGURE 4.1

B. SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what



other

FIGURE 4.2

C. USE CASE DIAGRAM:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

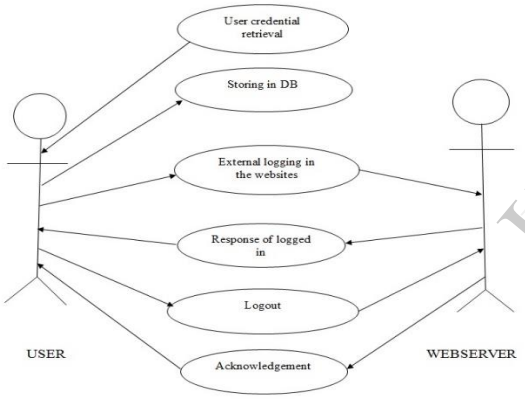


FIGURE 6.3

D. ACTIVITY DIAGRAM:

Activity diagram describe the workflow behaviour of a system. Activity diagram are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed.

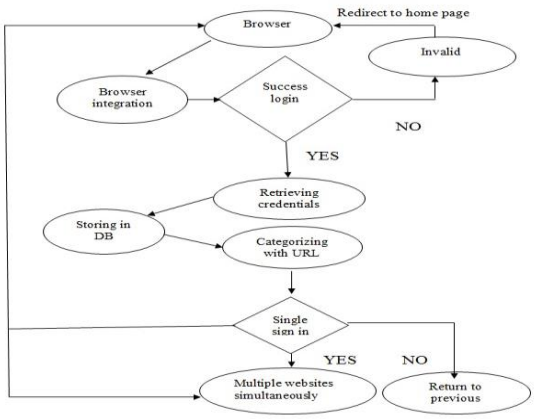


FIGURE 4.4

V. PROJECT DESCRIPTION

A. PROBLEM DEFINITION

The implementation of an SSO system has considerable positive aspects. First, navigating a website becomes exceptionally convenient because the single login will grant users access to nearly the whole site without any interruption for user authentication. Next, for the Network manager or website webmaster, there will be much less code to maintain, preserving site bandwidth and efficiency. Also, with permission via SSO, remote access is effortlessly possible; one example would be in accessing any nonlocal computer's (e.g., a website's server) files with your username and password. Finally, the same SSO connotation applies to the inverse Single Sign Out; this refers to the single action required to log out of the entire network of Programs and services of a website, usually by hitting the log out button.

As convenient as the SSO interface can be for both the client and manager side of networks, it presents noteworthy security risk as well. When clients use weak passwords and/or the same access credentials for different networks, their security is significantly diminished: malicious software deviant programmers and bad-bots can attempt brute force attacks to both client and manager accounts to gain access to their sensitive information. These attacks can be unrelenting and last for hours, attempting every imaginable username and password combination conceivable, usually with high success on freeware blogs and social website accounts, for example. In addition to the fact that if a network were to experience a failure, all users would effectively be locked out or worse, their sensitive information would be exposed for any given amount of time on an unsecured database. To overcome the more time consumption for multiple logging in we are proceeding this project to satisfy the simultaneous sign on for multiple websites using multiple credentials with the single click.

B. OVERVIEW OF PROJECT

1. SINGLE CLICK SIGN-ON (SSO)

Single sign On (SSO) (also known as Enterprise Single Sign On or "ESSO") is the ability for a user to enter the same id and password to logon to multiple applications within an enterprise. As passwords are the least Secure authentication mechanism, single sign on has now become known as reduced sign on (RSO) since more than one type of authentication mechanism is used according to enterprise risk models. For example, in an enterprise using SSO Software, the user logs on with their id and password.

This gains them access to low risk information and multiple applications such as the enterprise portal. However, when the user tries to access higher risk applications and information, like a pay roll system, the single sign on software requires them to use a stronger form of authentication. This may include digital certificates, security tokens, smart cards, biometrics or combinations thereof. Single sign on can also take place between enterprises using federated authentication. For example, a business partner's employee may successfully log on to their enterprise system.

When they click on a link to your enterprise's application, the business partner's single sign on system will provide a security assertion token to your enterprise using a protocol like SAML, Liberty Alliance, WS Federation or Shibboleth. Your enterprise's SSO software receives the token, checks it, and then allows the business partner's employee to access your enterprise application without having to sign on. Single sign on federated authentication also works with your employees. For example, an employee who is trying to access your outsourced benefits supplier to update their benefits information would click on the benefits link on your intranet. Your enterprise's single sign on software would then send a security assertion token to the benefits supplier. The benefits supplier's SSO system would then take the token, check it and grant access to your employee without making them sign on.

In today's world of complex distributed system, we see a lot of applications being used by organizations. Due to the increasing number of applications, management of user information has become a nightmare. Hence, a terminology called Single Sign on (SSO) was introduced. As the term suggests, it is a single sign-on for all the applications in an organization. A sign in to one is considered the same as a sign in to all. Logging in to one would mean a login to all.
For SSO to be implemented there needs to be a centralized authentication scheme. This scheme authenticates the user according to its own data store (which may be a database, LDAP, or any kind of password repository). It is responsible for signing in the users to all their associated applications/systems after authentication so that the user need not log in to each application individually. This centralized authentication scheme should also logout the user from all the applications on a single user logout.

2. INPUT DESIGN

The Design Input items are documented on the Design plan, and where ambiguity exists, will be clarified with the Customer and documented.

Design input may consist of:

- National/ International Codes of Practice,
- Customer supplied documents, drawings, specifications, samples,
- Statutory Regulations,
- previous similar designs,

This accumulation of design input information is reviewed by the Designer prior to commencing design process.

a. CLIENT-SIDE CREDENTIAL CACHING MECHANISM

The Secure Client-Side Credential Caching mechanism is a client-based SSO solution which keeps all the authentication information into a client-side credential storage. It allows the end-users to authenticate themselves once, and then has the system automatically provide the information for subsequent request without the users‟ intervention.

If the credentials are valid, the user will be authenticated transparently to the other application servers. This solution requires a high secure credential cache resides on client-side. It is very crucial to store the cached credentials securely as the credentials may be used to access some sensitive information or confidential web service.
So, it's not recommended to be used from portable client devices or some Operation Systems with a bad security reputation. Since all the authentication data is stored in the client-side credential cache, this architecture has little flexibility.

The user will get sign-on problems if he/she is not using his usual workstation (when travelling, for instance). Although it is relatively simple for the user to set up and configure, every time a new application server is added, the new authentication information should be added into the client-side credential cache.

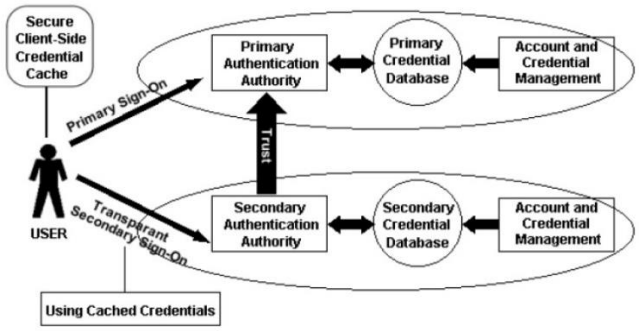


FIGURE 5.2.2.1

b. SERVER-SIDE CREDENTIAL CACHING BASED SSO

The Secure Server-Side Credential Caching mechanism is also called the server based SSO solution. The same as the Secure Client-side Credential Caching architecture, this approach also uses a central repository to store all the authentication information. But in this architecture, the cache is located on server side. It uses a central server to take on the task of administering all the different passwords and providing the needed information directly to the application asking for them

In a secure server-side credential caching mechanism, the primary credential database contains the user's primary credentials as well as the mappings between the primary credentials and the secondary credentials. The secondary credential database only keeps a copy of the secondary credentials. We should keep the mappings between these credential databases synchronized. There are three main approaches to achieve the data synchronization:

1. Integrating the credential synchronization services into the primary credential database.
2. Using external software to handle the credential synchronization process.

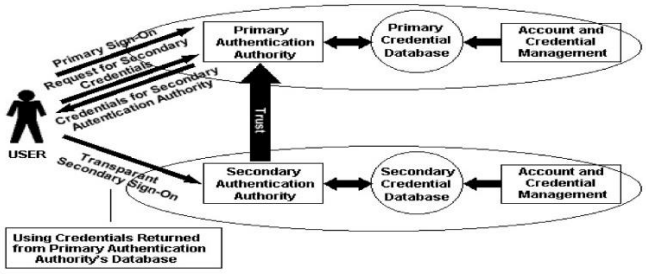


FIGURE 5.2.2.2

3. OBJECTIVES

Input design consists of developing specifications and procedures for data preparation, the steps necessary to put transaction data into a usable from for processing and data entry, the activity of data into the computer processing. The five objectives of input design are:

- Controlling the amount of input
- Avoiding delay
- Avoiding error in data
- Avoiding extra steps
- Keeping the process simple

4. OUTPUT DESIGN

The results of a design effort at each design phase and at the end of the total design effort. The finished design output is the basis for the Device Master Record. The total finished design output consists of the product, its packaging and labelling, and the Device Master Record

VI. CONCLUSION

It is an edict that seamless and transparent single sign-on multiple web sites without undermining overall network security and without requiring any online communications between service providers and the identity provider. It also allows for grained access control without any increase in the protocol's computational or communication complexity. Its simple access policy and user revocation management while providing nice forensic and audit data by building on common proxy signature strong and undeniable properties.

An authentication aw in the SAML SSO, can be generally exploited and reported. The featured quality of this"SINGLE CLICK SIGN ON MULTIPLE WEBSITES USING MULTIPLE CREDENTIALS" is to comfort the end user for fast log-in-out process with high security.

REFERNCES

1. A. Armando, R. Carbone, and L. Compagna. LTL Model Checking for Security Protocols. In *Journal of Applied Non-Classical Logics, special issue on Logic and Information Security*, pages 403–429. Hermes Lavoisier, 2009.
2. A. Armando, R. Carbone, L. Compagna, J. Cu´ellar, and M. L. Tobarra. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In *FMSE*. ACM, 2008.
3. A. Barth, C. Jackson, and J. C. Mitchell. Robust defenses for cross-site request forgery. In *15th ACM Conference on Computer and Communications Security (CCS 2008)*, 2008.
4. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *Advances in Cryptology CRYPTO 96*, volume 1109 of *LNCS*, pages 1–15. 1996.

5. Google. Web-based SAML-based SSO for Google Apps. http://code.google.com/apis/apps/sso/saml_reference_implementation_web.html, 2008.
6. T. Groß. Security analysis of the SAML Single Sign-on Browser/Artifact profile. In *Proc. 19th Annual Computer Security Applications Conference*. IEEE, Dec. 2003.
7. T. Groß, B. Pfitzmann, and A.-R. Sadeghi. Browser model for security analysis of browser-based protocols. In *ESORICS*, 2005.
8. S. M. Hansen, J. Skriver, and H. R. Nielson. Using static analysis to validate the SAML single sign-on protocol. In *WITS '05*, New York, NY, USA, 2005. ACM Press.