AN AUTHENTICATION AGENT FOR WEB-BASED SYSTEM

Wei Huang, Jinhua Xu

School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu, China
huangwei@suda.edu.cn, sujhxu@suda.edu.cn

Abstract

Web information systems are now widely used. These systems generally provide specific service in accordance with users' identities. To use these services, users are required to conduct identity authentication separately when logging in different web applications and systems. For the sake of security and access control, it is infeasible to use a unique common identifier and password for all systems. To achieve single sign on, we propose an authentication agent for web-based system which is called AA4WS. AA4WS can be installed in client side as a plug-in. It gets user's POST data when first time logging in some web application system, stores the data, interacts with the authentication agent server through web service, and simulates login procedure, therefore eliminating user's manual identity authorization and achieving unified identity authorization. AA4WS also achieves security in information storing, conveying, processing, updating and managing, and obtains efficiency as well, without adding additional service response time. AA4WS communicates with authentication agent server via web service by SOAP, making the system have characteristics of excellent generality.

Keywords: authentication; single sign on; access control; web service; plug in

1 Introduction

At present, web information system is widely used. Enterprises, faculties, institutes and organizations have their own web information management systems and information portals for various applications. These systems generally provide specific service according to users' roles. To take advantage of these services effectively, users have to remember accounts and passwords of each system. Many have been aware of its inconvenience. Some have provided ways to alleviate or solve it. For example, Microsoft provides functionality called autocompletion which is now widely used. With autocompletion, account and password typed by a user for a specific URI can be automatically stored. When the page is visited again, the password can be automatically filled by system as the account is typed. Despite of convenience brought by autocompletion, security will be a problem for Internet users. Hereby we put up a scheme aiming at 'logging in once, running everywhere', by which we develop an authorization agent for web-based system, called AA4WS.

2 Overview of AA4WS System

Although identity authorization systems are different in various web applications, most of them taking a HTTP POST approach. It submits a group of related information to authorization component for processing, usually by an ACTION component in a FORM. No matter what information is required by login page, POST format is constantly HTTP header plus name/value pairs. When the name/value pair is received, a server deals with validation according to user defined processing logic.

Hence, we can record HTTP POST data when a user logs in a system. When the user visits the page again, his/her request will be intercepted by the system, and then compose HTTP POST data after necessary processing. The data, afterwards, are sent to authorization components.

When a user visits a web system using Internet explorer, the system URI is send to authentication agent server after being captured by a monitoring plug-in installed in Internet explorer client. The authentication agent server distills HTTP POST data, HTTP header data and authorization URI. After that, the plug-in component rebuilds the data and constructs them to HTTP POST compatible data which will be sent to authorization component for users to log in.

The system mainly consists of monitoring plug-in, authentication agent server, and user information database. Plug-in is a COM module which derives from interface IDispatch and implements interface IObjectWithSite. It intercepts the system URI, interacts with the authentication agent server through web service, and simulates login procedure. Authentication agent server is designed as a web service, which provides identity authorization, agent URIs adding and login data acquiring. User

978-1-4244-4900-2/09/$25.00 ©2009 IEEE

information database stores important user information.

3 Design of Database for AA4WS

User information database contains important user messages, such as account, role and password, authentication token, and agent URI, which are particularly critical. We here show some key tables of database for AA4WS.

3.1 Table UserInfo

Table UserInfo is used to record basic information of users. The structure of the table is shown in Table I.

TABLE I UserInfo

Field	Comment
Account ID	Unique identifier of user
Account Name	User name
Password	Password for users to log in
Role	User role in the system

Password is stored in an encoded style to make sure that nobody can see the user's password. In AA4WS system, we use MD5 to encrypt the password. Therefore, when a user logs in, his/her password should be encoded by MD5 before comparing with password stored in database.

Authentication agent server returns an authentication token after validation using account and password.

3.2 Table Token

Table Token is used to store user token information. The structure of the table is shown in Table II.

TABLE II TOKEN

Field	Comment
Token ID	Identifier for token which is a random string
Account ID	Unique identifier of user
Current Machine	Current logging in machine
Expired Time	Expired time of user session

Authentication token of AA4WS server is similar to session in web service. Usually, a server will invalidate token automatically if the user doesn't use token to access authentication agent server in a period of time, e.g. 20 minutes. Once a user is validated, other system registered in AA4WS can use the token to communicate with the server. Authentication token is shared by all AA4WS in user's computer, which is essentially different from session.

3.3 Table URIAgent

Table URIAgent is used to save user agent data. After validation, authentication agent server will get POST data, HTTP header, referred URI and authentication page URI from table URIAgent. The structure of the table is shown in Table III.

TABLE III URIAgent

Field	Comment
Account ID	Unique identifier of user
URI	URI of the system for user to log in
POST	HTTP POST data
HEADER	HTTP header data
Reference URI	Reference URI
Authentication URI	URI for authentication

As URI, POST, HEADER, Reference URI and Authentication URI in URIAgent are critical to users, they should be encrypted by AA4WS before being stored in the database.

4 Design of Authentication Agent Server

AA4WS is a plug-in which can be used in Microsoft Internet Explorer and Microsoft Windows browser, as well as Internet explorers and browsers developed by other venders. To attain as much generality as possible, AA4WS interacts with authentication agent server through web service by SOAP. Therefore, network communication security and service response time will be particularly critical considerations for authentication agent server. We will discuss the issues between network communication security and service response time via analyzing several key services of authentication agent server.

4.1 User Login Service

User login service provides identity authentication to log into authentication agent server. We process user login as follows.

Step 1 AA4WS uses HTTPS to connect with authentication agent server and then sends account and password to server via SOAP;

Step 2 The server receives account and password from AA4WS;

Step 3 Encrypt password before comparing with the password in database;

Step 4 After validation, AA4WS gets URI data from table URIAgent according to user account identifier;

Step 5 AA4WS sends an authentication token to user;

Step 6 Get data from AA4WS and form cache list;

Step 7 Insert a new record into table Token.

After logging in, an URI cache list is sent back to AA4WS, therefore AA4WS will know which URI uses agent without accessing authentication agent server. As a result, the browsing speed will not be affected by AA4WS.

4.2 URI Adding Service

We can insert a new agent URI into agent authentication server. The processing logic is given as follows.

Step 1 AA4WS connects with agent authentication server using HTTP protocol;

Step 2 Server receives authentication token and encrypted URI, POST, HEADER, Reference URI and Authentication URI data from AA4WS;

Step 3 Authentication server gets Account ID by authentication token;

Step 4 Set Expired Time in token as current server time plus existing time selected by the user;

Step 5 Authentication server inserts Account ID, URI, POST, HEADER, Reference URI and Authentication URI into table URIAgent.

After AA4WS has saved POST data sent at the time of accessing web, the data are encrypted whose key is produced by user login password and then sent to authentication agent server to form a new record.

4.3 URI Data Acquiring Service

AA4WS gets POST, HEADER, Reference URI and Authentication URI data of agent URI from authentication agent server when predicting the URI could be agented.

Step 1 AA4WS connects with agent authentication server using HTTP;

Step 2 Server receives authentication token and encrypted URI address from AA4WS;

Step 3 Authentication server gets Account ID by authentication token;

Step 4 Set Expired Time in token as current server time plus existing time selected by the user.

The POST, HEADER, Reference URI and Authentication URI data of agent URI are acquired according to Account ID and encrypted URI address.

4.4 Cache URI Acquiring Service

When sensing that current URI is not in Cache list, AA4WS discharges current request directly and creates a new thread to fresh Cache list in server.

Step 1 AA4WS connects with agent authentication server using HTTP protocol;

Step 2 Server receives authentication token from AA4WS;

Step 3 Authentication server acquires Account ID according to authentication token;

Step 4 Take URI value from table URIAgent and then send it to AA4WS;

Step 5 Set Expired Time in token as current server time plus existing time selected by the user.

AA4WS creates a new thread in the server to refresh so as to ensure the synchrony between URI Cache lists of current URI, meanwhile protecting from affects browsing speed.

4.5 Account Management Service

Account can be managed by users. Users can apply to the authentication agent server to add a new account, modify password, or manage an existing agent URI.

Step 1 Server receives account, previous password, new password and confirmation password;

Step 2 Server determines whether it is correct according to account and previous password;

Step 3 After validation, the server decides whether new password is consistent with confirmation one;

Step 4 Server acquires URI, POST, HEADER, Reference URI and Authentication URI data from table URIAgent by Account ID, deciphers the data, encrypts the data using new key from new password, and finally stores them into table URIAgent.

5 Conclusions

Web information systems are widely used presently. Users are required to input account identifier and password pairs when logging in different web applications and systems, which is quite tedious. Unfortunately, due to security consideration, it is infeasible to make all systems share a common account and password.

We propose an authentication agent for web-based system solving the problem of repeat identity authorization when a user logs in different Web application systems. We achieve security in information storing, conveying, processing, updating and managing, and obtain efficiency as well, solving the contradiction between service communication security and service response time.

AA4WS takes a client plug-in approach. It gets user's POST data when first time logging in some web application system and stores the data in database. AA4WS will automatically distill posted data from database when the user logs in again, thus eliminating user's manual identity authorization, achieving unified identity authorization. AA4WS communicates with authentication agent server via web service by SOAP. As a result, the system has characteristics of excellent generality.

References

[1] Fumiko Satoh, Takayuki Itoh. Single Sign On Architecture with Dynamic Tokens. *2004 Symposium on Applications and the Internet (SAINT'04)*, 2004.

[2] J.S. Park, R. Sandhu. Binding identities and attributes using digitally signed certificates. *16th Annual Computer Security Applications Conference (ACSAC'00)*, 2000.

[3] Fang Ying-lan, Han Bing, Li Ye-bai. Research and Implementation of Key Technology Based on Internet Encryption and Authentication. *2009 International Conference on Networking and Digital Society*, vol. 1, pp.179-182, 2009.

[4] Yan-zhi Hu, Da-wei Ma, Xiao-fei Li. An Improved Authentication Protocol with Less Delay for UMTS Mobile Networks. *2009 International Conference on Networking and Digital Society*, vol. 2, pp.111-115, 2009.

[5] Jeremy Goold, Mark Clement. Improving Routing Security Using a Decentralized Public Key Distribution Algorithm. *Second International Conference on Internet Monitoring and Protection*, 2007.

[6] Jian Liu, Dianfu Ma, Zhuqing Li, Dou Sun. A Formal Description of Web Services Container Architecture. *2009 Fourth International Conference on Internet and Web Applications and Services*, pp.30-36, 2009.

[7] Jingjun Zhang, Fanxin Meng, Guangyuan Liu. Research on SOA-Based Applications Based on AOP and Web Services. *2008 International Conference on Computer and Electrical Engineering*, pp.753-757, 2008.

[8] Youxiang Duan, Yongtang Bao, Lijiang Pan, Beibei Yan, Jiuyun Xun, Nianyun Shi. A Secure Web Services Model Based on the Combination of SOAP Registration Info and Token Proxy. *2008 International Symposium on Computer Science and Computational Technology*, vol. 2, pp.15-20, 2008.

[9] Jian Wu, Zhimin Huang. Proxy-Based Web Service Security. *2008 IEEE Asia-Pacific Services Computing Conference*, pp.1282-1288, 2008.

[10] Udaya Kiran Tupakula, Vijay Varadharajan, Sunil Kumar Vuppala. SBAC: Service Based Access Control. *2009 14th IEEE International Conference on Engineering of Complex Computer Systems*, pp.202-209, 2009.

[11] Sebastian Rieger. User-Centric Identity Management in Heterogeneous Federations. *2009 Fourth International Conference on Internet and Web Applications and Services*, pp.527-532, 2009.

[12] Keith Williamson, Michael Healy. Industrial Applications of Software Synthesis via Category Theory. *14th IEEE International Conference on Automated Software Engineering (ASE'99)*, 1999.

[13] Bhosale, S.K. Architecture of a single sign on (SSO) for internet banking Wireless, Mobile and Multimedia Networks. *IET International Conference*, 2008.

[14] T. Groß. Security Analysis of the SAML Single Sign-on Browser/Artifact Profile. *Proc. Ann. Computer Security Applications Conf. (ACSAC)*, 2003.

[15] Vipin Samar. Single Sign-On Using Cookies for Web Applications. *IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1999.