Web Services Based Authentication System for E-Learning

Akram Alkouz[1] and Samir A. El-Seoud[2]
[1]Computer Graphics and Animation Department, [2]Computer Science Department
Princess Sumaya University for Technology (PSUT) - Jordan

Abstract: *In Distance Learning end users need to access different e-learning platforms daily to gain the knowledge. E-learning platforms implement authentication system to handle the authentication and authorization processes. As the number of directory stores grows the development overhead of user's authentication process in e-learning platforms against those directories increases. Also as the number of e-learning platforms grows, the number of user's IDs and passwords users have to memorize grows as well. So users make passwords not strong enough to ease memorization, and write passwords in clear text in insecure places, which compromise the security. An outline of various aspects of design and implementation of web services based authentication system for e-learning platforms (WSAS) is presented in this paper. The architecture provides e-learning platforms users with a single sign-on solution for the problem of memorizing many user IDs and passwords, provides organizations with a centralized, simple, and efficient directory stores access mechanism to simplify the process of integrating multiple directory stores, and provides the e-learning platforms developers with a standard solution to minimize the development overhead of the authentication process against multiple directory stores, the presented prototype architecture designed based on the existing web services technology, so that clients need not be modified, and servers may have a little modifications.*

Key words: *Single-sign-on, web services, authentication, e-learning*

Received: January 31, 2007 | ***Revised:*** June 15, 2007 | ***Accepted:*** August 25, 2007

1. Introduction

E-learning platforms are heterogeneous environments where you can find different web enabled applications have high level of interoperability between each other running on different operating systems. Also these applications use different authentication systems. Authentication system verifies the Identity of the user against directory stores. Where directory stores are databases of usernames, passwords, and profile information of every user on the network, and it can be relational databases, directory servers, or text files. E-learning platforms face these challenges, 1) each application needs to implement its own authenticate process against its directory store, 2) As the number of directory stores grows, the development overhead will increase, 3) As the number of e-learning platforms grows, it becomes hard for users to remember IDs and passwords. In this paper a solution that can handle the three mentioned challenges will be presented. However, different solutions that can handle some of these challenges do exist, in this paper explanation to some of them will be presented.

The presented solution WSAS is flexible in terms of integration to heterogeneous platforms and easy to deploy because it based on the standard technologies of Web Services such as XML, WSDL, UDDI, and SOAP. Web service is application logic accessible to programs via standard web protocols in a platform-independent way [2]. It can help in implementing a solution that can handle the three mentioned challenges. The single-sign-on mechanism of WSAS that can authenticate users against multiple directory stores in a secure, unified, and centralized way will be investigated

The most common authentication systems are, SSL, Kerberos, and Microsoft Passport will be explained. SSL is part of the established web infrastructure, and provides confidentiality and peer authentication, but in its established form it does not offer single sign-on functionality [11]. Kerberos, in turn, provides a

solution to the problem of authentication, key exchange, and single sign-on [4], but it can't deal with multiple directory stores [10]. Added to that that it cannot be used for internet single sign-on environment, where plug-ins is not allowed. Microsoft Passport provides a solution, but it cannot deal with multiple directory stores and it has some key management problems [5].

2. System Architecture

WSAS system consists of four tiers as in Figure 1, User, Application Server, Master Server, and Directory Stores tier. In this section the role of each tier will be described.

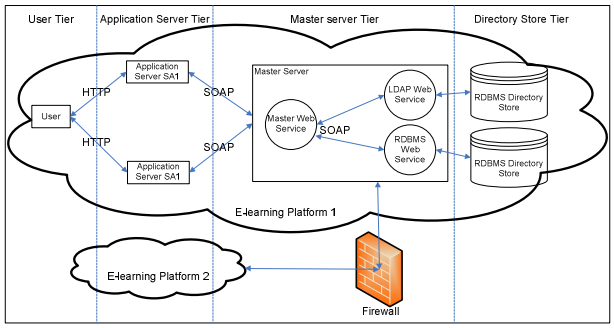

Figure 1. WSAS architecture with e-learning platforms integration through firewall.

User tier is the interface for the user in the e-learning platform that has access to one or more of the Application Servers. Users usually use browsers to access web enabled applications hosted on Application Servers. The crucial property of the WSAS system is the role of the user's web browser as a relay of user-specific information, delivering messages between the application servers and the Master Server. This is accomplished by means of ordinary WWW technologies, HTTP redirects, URL query strings, and cookies [6]. Browser delivers encrypted authentication information with every request, and thus the application server needs to check the validity of the authentication data with the Master Server. User's browser acquires this authentication information, in form of an *AuthTicket* stored in cookie [3]. Browser's role as a message relay has its limitations. It is inefficient to deliver large amounts of data due to bandwidth considerations [7]. However, the application server can perform authentication of the user in place, the latency of the authentication, once the user has acquired the AuthTicket, is minimal.

Users connect to Application Servers tier using HTTP messages. Users need to be authenticated to the application server. Based on the result of the authentication the application server can grant the users permissions to access different resources on the server according to each user privileges. Granting privileges is part of the authorization process which is specific to each e-learning platform.

Master Server tier is the core of WSAS system, It acts as a trusted third party server among e-learning platforms. Master Server handles the process of authenticating users, integrating multiple directory stores into WSAS system, and providing the integrity, authenticity, and confidentiality of the messages passed between users and application servers. Master Server built based on the web services technology, where web services can provide centralized and unified way to build the authentication process, also it makes it easy to build distributed and scalable components to interface with different directory stores. Since web services is application logic that can be accessed via standard protocols SOAP in platform independent way [8], that means web services can provide a solution to authenticate users and check the validity and authenticity of users for many different platforms. Master Server is an application server that hosts three sets of Web Services, Master Web Service, LDAP Web Service, and RDBMS Web Services, as in Figure 2.
Master Web Service will handle all user's requests for authentication redirected by different application servers, receive the redirected request, check the available directory stores within WSAS, and based on that it will issue asynchronous method call to the LDAP or RDBMS web services using SOAP protocol, and waits for any of them to response with positive results. Also it will handle the process of integrating new directory stores into WSAS. New e-learning platforms need to participate in WSAS will use the Master Web Service WSDL file to be able to access WSAS.

LDAP and RDBMS Web Services will handle the process of authenticating users against the available LDAP or RDBMS directory stores. By receiving the asynchronous call from the Master Web Service, LDAP or RDBMS web services will issue asynchronous call for the available LDAP or RDVMS directory stores, if

any of them response with positive results it will kill the other calls and return back to Master Web Service.

Directory Stores tier represent a database that used to store the user's profiles, it can be RDBMS, LDAP directory stores, or a text file. Most common range of these directory stores are supported within WSAS. It can be accessed based on the configuration information posted to the Master Server. New types

of directory stores can be easily integrated to WSAS, by building a new web service that can interface with these new directory stores, and deploying it into the Master Server with the appropriate configurations to be able to communicate with Master Web Service.

Different e-learning platforms can be easily integrated to WSAS system, where the Master Server is build based on web services technology which is firewalls friendly because methods can be called by SOAP protocol, which does not need open new ports in the firewall, where SOAP encapsulated over HTTP. E-learning platforms can benefit from WSAS by either using the Master Server to authenticate users or by deploying the service to their own application servers, as in Figure 2.

3. Methodology

In this section how WSAS works will be described, the three aspects of system work: deployment, development, and user sign-in process and messages flow will be described in details.

Deployment of WSAS requires ISS server with Microsoft.Net framework, after deployment WSAS needs to know the current directory stores used in the e-learning platform, so the system administrator needs to supply the connectivity string of directory stores. Directory stores information stored in XML file that will be used later to connect to the directory store.

WSAS development requires the developers to get the WSDL file online from the Master Server, and based on the WSDL file build the proxy file specific to each platform, where for Java platforms we need to build Java proxy file, for .Net platform we need to build ASP.Net proxy file. Proxy file acts as interface between the Application Server and the Master Server. Using proxy, web services can become just as integral to our applications. Instead of the logic within the methods doing simple file I/O, or local machine or network functions, the same black-boxed functions can call web services methods anywhere , our application neither knows nor cares where the data comes from, nor the logic behind it. When the developer builds the proxy class, the tools use the WSDL that defines our web service to generate the appropriate methods, with related data types intact. With the proxy already built, the web service consumer simply calls the web method from it, and the proxy, in turn, performs the actual request of the web service. When we refer the web service in the consumer application, it appears to be part of the consumer application, like a normal internal function.

When the users use the system for authentication, two scenarios are expected, first scenario where the user wants to access Application Server SA1, and second scenario user wants to access Application Server SA2, as in Figure 4.

First Scenario: User Wants to Access Application Server SA1

User tries to access SA1 application server, as in Figure 4. If user does not has a

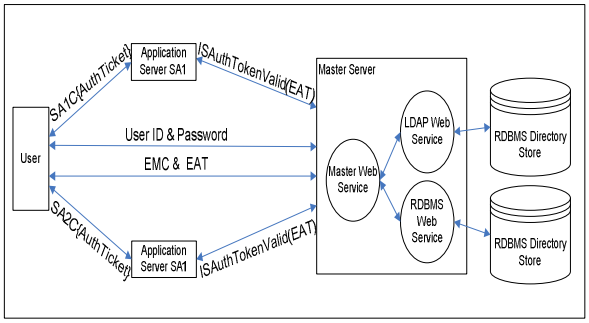

Figure 2. User sign-in process, user Accessing SA1.

valid (AuthTicket) cookie, SA1 will detect that the user is not authenticated and will redirect the user to the Master Server**.** Master Server asks the user for his credentials, user will submit his credentials to the Master Web Service on the Master Server. Master Web Service will read the XML file that contains the directory stores information, For each directory store, Master Web Service will communicate with LDAP and RDBMS Web Services via asynchronous web method to validate user credentials against directory stores, if it is valid user Master Server will create encrypted master

cookie (EMC) in user's browser, and redirect the user back to SA1 with Encrypted Authentication Token (EAT) included in the redirected message. SA1 will get the EAT from the query string, and check the authenticity of EAT by invoking IsAuthTokenValid() web method in the Master Web Service. If the authenticity check result is OK, SA1 will create encrypted cookie (SA1C) in the user's browser. When the user returns back to SA1, the EAT will returns as well, so SA1 can detect that the user is already authenticated.

Second Scenario: User Wants to Access Application Server SA2

User tries to access SA2 application server as in Figure 4. If user does not has a valid AuthTicket cookie, SA2 server will redirect the user to the Master Server, EMC is sent to Master Server, Master server can detect this user is already authenticated. Master Server will redirect the user back to SA2 with EAT included in the redirected message. SA2 will get the EAT from the query string, and check the authenticity of EAT by invoking IsAuthTokenValid() web method in the Master Web Services. If the authenticity check result is OK, SA2 will create encrypted cookie SA2C in the user's browser. When the user returns back to SA2, the EAT will returns as well, thus SA2 Server can detect that the user is already authenticated.

WSAS as a system offers a secure way of providing user's authentication, WSAS relating to cryptographic services as follows: *Confidentiality* of the exchanged messages is preserved: the information is encrypted using symmetric block cipher Triple-DES (3DES,), and only the entities possessing the key are able to read the information. Keys are generated on fly, and exchanged online by means of WS-Security [12]. *Message integrity* is preserved in the WSAS system using cryptographic SHA-1 digests of the exchanged information which is based on the WS-Security specification and WSE implementation [1]. *Authentication* is the main services provided by the WSAS, and it is achieved using WSAS Master Server, which acts as a Trusted Third Party.

The WSAS has different ways to log out from the system. The Encrypted Master Cookie (EMC) and server specific cookie such as SA1C have a limited lifetime. If user's EMC has expired, the user is logged out. Since the cookies used by the WSAS are by default session cookies, a simple way to log out of the system is to close the browser. WSAS also provides a GUI mechanism to logout without closing the Browser.

4. Conclusion

The research problem and goal was to design and implement a feasible solution based on web services for user authentication, and based on that, to design a controlled and secure method of accessing different user's directory stores for delivery and distribution of single sing-on service within e-learning platforms.

The design and implementation of WSAS was investigated. WSAS provides e-learning platforms developers with a unified and centralized authentication process that can be accessed from different platforms. The authentication process build based on web service technology, which enables application servers running on different platforms to integrate to the system in an easy way. WSAS provides e-learning platforms users with a cross platform Single-Sign-On authentication system for the problem of memorizing many user's IDs and passwords. Adding new application servers to the WSAS is a matter of building proxy file to consume the web service. Adding directory stores with the LDAP or RDBMS category is a mater of a configuration file, while adding a new category of directory stores is as easy as building the logic of interfacing the new directory store into a web service and deploying the web service into the Master Server. Secure transfer of authentication XML and HTTP messages between User, Application Servers, and Master Server is achieved. WSAS should function with user's existing browser, without additional plug-ins in usual network configurations. Using web service makes it very easy to integrate many e-learning platforms to the system, where web service exchanges messages by using SAOP over HTTP, so there is no need to open new ports on the firewalls. The nature of web services make it easy to deploy the Master Server on one operating system and the application servers on different operating system, while keeping the message exchanged between the Master Server and Application Servers secure. exchanging public keys on fly in secure way using WS-Security reduce the headache of key management in PKI environments. Using master key to generate

specific key for each user to encrypt his cookies make it too difficult for attackers to compromise the security by attacking someone cookies.

Using Microsoft WSE to encrypt the message between the Master Server and the Application Servers is the main drawback of WSAS. However Microsoft WSE is the only available implementation of WS-Security.

WSE is only supported on .Net framework [9], so Application Servers on other platforms can't benefit form WSE to secure the exchanged messages with the Master Server, but still they can integrate and use the authentication service provided by WSAS, and keep the security by using SSL connection between the Master Server and the Application Servers.

5. References

[1] Atkinson Bob, Della-Libera Giovanni, Hada Satoshi, "*Web Services Security (WS-Security) Version 1.0 05*", Microsoft corp., April 2002.

[2] Basiura Russ, Batongbacal Mike, Bohling Bendon, *Professional ASP.Net Web Services*, Wrox, 2001.

[3] Fielding Roy T., Gettys James, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, Tim Berners-Lee, "*Hypertext Transfer Protocol HTTP/1.1. RFC 2616*", IETF Network Working Group, June 1999. http://www.ietf.org/rfc/rfc2616.txt

[4] John T. Kohl, B. Neuman Clifford , "*The Kerberos Network Authentication Service (V5) RFC 1510*", IETF Network Working Group, September 1993. http://www.ietf.org/rfc/rfc1510.txt 51-58, 2000.

[5] Korman David P., Rubin Aviel D., "*Risks of the Passport Single Signon Protocol*", Computer Networks, Elsevier Press, volume 33, pages 15 http://www.ietf.org/rfc/rfc2616.txt

[6] Kristol David M., Montulli Lou, "*HTTP State Management Mechanism. RFC 2965*", IETF Network Working Group, October 2000. http://www.ietf.org/rfc/rfc2965.txt

[7] Kristol David M., "*HTTP Cookies: Standards, Privacy, and Politics*", ACM Transactions on Internet Technology (TOIT), November 2001.

[8] Microsoft corp., *Web Services Documentation*, http://msdn.microsoft.com/webservices/understanding/webservicebasics/default.aspx?pull=/library/en-us/dnwebsrv/html/webservbasics.asp http://arxiv.org/PS_cache/cs/pdf/0105/0105018.pdf

[9] Microsoft corp., "*WS-Security Specifications*" http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnglobspec/html/wssecurspecindex.asp

[10] Schneier Bruce, *Applied Cryptography*, John Wiley and Sons, 1996.

[11] Thomas Stephen, *SSL and TLS Essential*, John Wiley & Sons, 2000.

[12] Zilinskas Adam, Brown Morris, Loomis Brian, *Securing B2B XML web Services with WS*, Microsoft corp., April 2002.

Professor Samir Abou El-Seoud received his BSc degree in Physics, Electronics and Mathematics from Cairo University in 1967, his Higher Diplom in Computing from Technical University of Darmstadt (TUD) /Germany in 1975 and his Doctor of Science from the same University (TUD) in 1979. Professor El-Seoud helds different academic positions at TUD Germany. Letest Full-Professor in 1987. Outside Germany Professor El-Seoud spent different years as a Full-Professor of Computer Science at SQU – Oman and Qatar University and acted as a Head of Computer Science for many years. At industrial institutions, Professor El-Seoud worked as Scientific Advisor and Consultant for the GTZ in Germany and was responsible for establishing a postgraduate program leading to M.Sc. degree in Computations at Colombo University / Sri-Lanka (2001 – 2003). He also worked as Application Consultant at Automatic Data Processing Inc., Division Network Services in Frankfurt/Germany (1979 – 1980). Professor El-Seoud joined PSUT in 2004.

Akram Alkouz received his BSc degree in Computer Science from Princess Sumaya University for Technology /Jordan 1996, his Master Degree in Computer Science and Information Engineering from National Chiao Tung University Taiwan in 2003. Akram Alkouz joined PSUT since 2003