Using OpenID for E-learning Applications

FELICIAN ALECU[1], PAUL POCATILU[1], GEORGE STOICA[2], CRISTIAN CIUREA[1], SERGIU CAPISIZU[3]
[1]Economic Informatics Department, Academy of Economic Studies
Pta. Romana 6, sector 1, Bucharest
[2] ProSoft Solutions,
[3]Bucharest Bar Association
Bucharest
ROMANIA
alecu.felician@ie.ase.ro, ppaul@ase.ro, george.stoica@pss.ro,
cristian.ciurea@ie.ase.ro, capisizu@mb.euroweb.ro

Abstract: Complex distributed e-learning applications require special focus on security. Distributed e-learning applications have several modules user can access using different clients (desktop or mobile). The same user has several accounts with different credentials. Using OpenID standard for e-learning Web-based applications is a good solution since the users need to access various areas and modules of the application. This paper presents the main characteristics of OpenID standard and how this standard could be implemented for a distributed, Web-based, e-learning application.

Key-Words: OpenID, Single Sign On, security, e-learning solutions, mobile applications.

1 Introduction

In many fields, the security is the most important quality characteristic of an application.
Web-based applications are exposed to many attacks and it is less expensive to pay hackers to discover the vulnerabilities than to launch in real environment an application that is not tested enough.
In order to ensure the high security level inside theirs information systems, many organizations engaged real hackers to test and discover the vulnerabilities of every new application that will be launched in production.
The same applies for the security of distributed learning applications that is very important and challenging compared with standalone applications.
There are several areas where the security requirements are high and they need special attention. These issues can be managed using several methods and techniques like:
- different authentication levels;
- password management;
- data encryption;
- location services.

Each of actions from Table 1 requires a certain degree of security, depending on the importance and data sensitivity.
The databases with tests, marks and users contain sensitive data and they need a special attention.
The security requirements for examinations, homework/project assessment and user management are very high due to the importance of data and information they use. To increase the users' responsibility concerning the data introduced in the database of the e-learning

application, the password of every user must be encrypted, so that nobody can read it, even the application administrator.
The specific features of e-learning applications have represented criteria in choosing the best encryption algorithm.
In [6] is considered that the RSA, DES, MD5, SHA, Blowfish, Diffie-Hellman, ElGamal, and AES encryption algorithms are the most efficient in the open source applications and have good implementations.
Quizzes and content management have medium security requirements, because the data manipulated is less sensitive.
Feedback management and messaging for these systems does not use sensitive data.

Action	Security requirements
Online exams	High
Content management	Medium-High
Feedback management (forums)	Low-Medium
Homework/Projects Assessment	High
Quizzes	Medium
User management	High

Table 1. Security concerns of m-learning applications

Table 1 presents some security concerns regarding the mobile learning solutions, based on [7].
The minimum requirement is to use authentication through users and passwords. In order to increase the security of e-learning applications, these must be protected against SQL injection, so that only authorized users can access them. The protection against SQL injection is usually realized by minimizing the letters entered by users in the textboxes for username and password and by replacing the special characters associated with an SQL statement.
Wireless data communication can be easily monitored, so high security need to be assured by using specific standards. Application testing has to include several security tests.

2 OpenID - Necessity

Internet applications have always presented major security vulnerabilities. The most important method of security is authentication and authorization of users, but considering that the main model for establishing identity is based on providing a username and password, this can easily determine that the authentication process is the main "bridgehead" for a possible computer attack. Solutions to this problem include the use of SSL protocol that is less vulnerable. But a new problem arises. In the context of developing Web 2.0 applications (blogs, wikis, social networks) and their utilization on a large scale, using several applications of this kind can affect the ease of use. In these circumstances, the use of such facilities requires an additional effort in managing credentials (different for each application). This is why it appears the necessity of a system that allows multiple authentications using a single set of credentials. In this way, the security of the entire process could be improved.
A solution to the problems described above could be the OpenID standard. Actually it is a protocol implemented by providers such Verisign, Yahoo, Google, Microsoft and many others. This solution enables users to use different authentication services based on digital identities, successfully implementing the concept of "single sign on" and ease of use [2].

3 Single Sign On Basics

As systems get into the business and actively participate in the development process, users and administrators have difficulty in managing the activities. Typically, users are forced to log on into multiple systems using different sets of passwords and user names. Administrators must manage user accounts so that they are accessed in a coordinated manner that does not affect the integrity of security policies.

The application administrator is dealing with the access rights of each user, adding and deleting some users according with the security rules.

The concept of Single Sign On allows the access control to independent systems. This method allows the user to authenticate once and gain access to all systems without needing a new login. The reversed process of the Single Sign On authentication is the Single Sign Off – running the logout procedure once the user is actually loosing any right of access to all the systems.

The most common Single Sign On configurations involve the use of Smart Cards, OTP tokens and Kerberos protocol.

Kerberos is the easiest way to utilize the Single Sign On concept since it is implemented in most operating systems currently in use,

There are actually three types of Single Sign

On services available [3]:

• Integrated Windows Single Sign On - allows connection of multiple network applications using a common authentication mechanism. An example is using the Kerberos protocol at the network level.

• Extranet Single Sign On (Web Single Sign On) - allows accessing Internet resources using a single set of credentials. Examples of this option are Open ID or Microsoft Passport Network.

• Server-Based Intranet Single Sign On - allows integration of several heterogeneous applications that do not necessarily use the same authentication mechanism.

• Enterprise Single Sign On - designed to minimize the number of authentications of a client in various applications. Provides ability to send encrypted user credentials across the network.

The following protocols are used in OpenID

implementations:

• Diffie - Hellman Cryptographic Protocol

• Secure Hash Algorithm

• Yadis

Diffie-Hellman was developed in 1976 and published in the article "New Directions in Cryptography" by Whitfield Diffie and Martin Hellman. It is the first practical method by which two distinct entities can establish a secret key using an insecure connection. It is using an asymmetric encryption algorithm so each entity owns a public and a private key. [8]

The original algorithm implementation involves the use of a multiplicative group *modulo p*, where p is a prime number and g is a primitive element *modulo p.*

The main vulnerability is an attack of the "man in the middle" type. The solution to this problem is the Diffie Hellman protocol with authentication known as Station-to-Station protocol (STS), developed by Diffie, van Oorschot and Wiener in 1992. Immunity is achieved by allowing the entities to authenticate using digital signatures and certificates.

Secure Hash Algorithm is a dispersion cryptographic function recommended by NIST [1], [9]. There are several versions of this algorithm:

• Secure Hash Algorithm - 1: The original version with a 160-bit output, developed by the NSA (National Security Agency).

• Secure Hash Algorithm - 2: Versions with output of 224, 256, 384 and 512 bits.

• Secure Hash Algorithm - 3: the

developing standard. Selecting the new algorithm will finish in 2012 at the end of the competition organized by NIST.
The result of the function is obtained at the end of an iteration consisting of 64/80 steps.
Dispersion functions offer the possibility to make certain the integrity and authenticity of messages. This is the main reason why these functions are used with digital signatures. Secure Hash Algorithm - 1 is used by the TLS, SSL, PGP, SSH, IPsec and DSS cryptographic algorithms.
Secure Hash Algorithms are vulnerable to a series of attacks, like the following:
• Finding the corresponding hash message by brute force ("preimage attack")
• Finding two messages that have the same hash (collisions).
• Attacks of "meet-in-the-middle" type that reduce the number of operations needed to "break" the function.
Yadis is a protocol used to discover web services like OpenID, OAuth and XDI connected to a Yadis ID. Although it was designed to discover authentication services, Yadis can easily be used with other types. The identifier may be a URL or an XRI that conducts to a URL. Using the protocol it is possible to obtain a descriptor of the service in the shape of a XRDS document. In OpenID, Yadis is used in the discovery phase of an authentication service provider.

4 OpenID Protocol – Overview

Currently, the primary means of user authentication in the online environment is the concept of "username-password". Disadvantages of such a solution are obvious when it is necessary to use several applications that require independently a set of credentials. OpenID protocol offers a possibility to avoid such situations by introducing the concept of ID. With this identifier recognized by providers of applications, a user can login once using the same set of credentials.
Entities participating in the login session are described below (figure 1):
• The end user who wants to access an application.
• OP (*OpenID Provider*) – Server that stores recorded OpenID identities and performs the authentication process for customer pages when receiving requests to do so. It presents a Web interface that allows users to create OpenID identities that can be assigned to one or more profiles. The provider has the responsibility to receive applications for customer login pages and to resolve these claims under OpenID specifications.
• RP (*Relying Party*) – any web application that implements OpenID authentication process. RP can ask OP the user details (name, e-mail) eliminating the need for registering an account on every website.

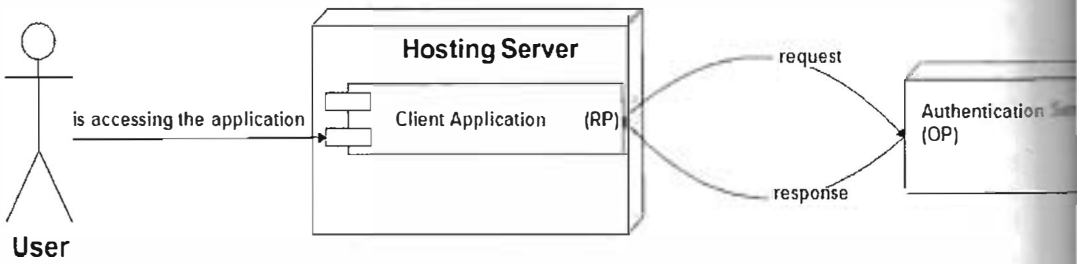

Figure 1 – OpenID Entities

The authentication process takes place in seven simple steps:
1. Initialization of authentication by presenting the user identifier
2. Right after normalization of the identifier, the client application tries to find the address of an authentication server.
3. (optionally) The client application establishes an "association" with the server using the Diffie-Hellman protocol.
4. The client application redirects the user to the server for authentication process.
5. The server determines if the user is authorized to access the client application.
6. The server redirects the user to the client application based on the response to the authentication request.
7. The client application verifies the information received from the server

OpenID protocol supports two operating modes: with and without state details (stateful/ stateless), also known as smart and dumb mode. The operating mode is decided by the response to the request of the association that is sent in the initial phase of the protocol. A negative response will determine the "dumb" mode activation – the messages from the server are checked by direct requests to PO by setting the value of the *openid.mode* parameter as "check- authentication". In "smart" mode, the message verification is performed by the RP using the details generated in the context of the association activity.

OpenID is using two types of requests, like the following:
- Direct requests - between RP and OP. An example of using direct applications is the checks of the messages coming from the server even if an association between RP and OP is not established.
- Indirect requests - messages go through the user's browser. Indirect communication is used in the authentication phase. There are two methods of indirect communication: HTTP redirects and HTML forms.

There are several OpenID providers like:
- ClaimID
- GetOpenID
- Google
- MyOpenID
- Orange
- VeriSign
- WordPress
- Yahoo

Also, there are open libraries that allow creating your own OpenID providers.

5 E-learning application architecture

In order to use OpenID for an e-learning application, this requires several changes. Figure 2 shows the interactions between the e-learning client and the e-learning server.

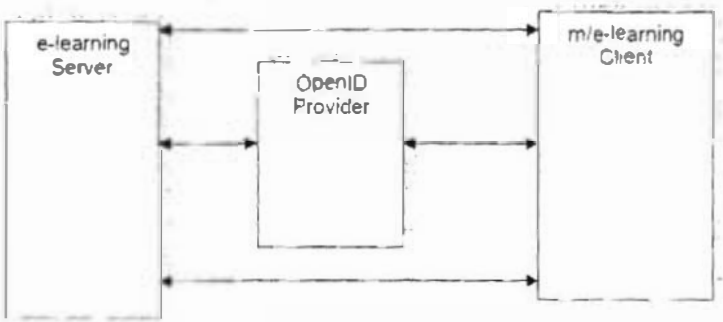

Figure 2 – Interaction between the e-learning client and the e-learning server

The client access the e-learning server using a desktop computer or a mobile device. The login window allows connecting directly, using server credentials to connect to OpenID provider. If the user chooses to connect through OpenID provider, the user enters his or her OpenID identity (figure 3).

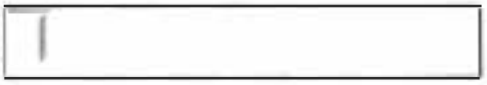

Figure 3 – OpenID identity text control

If the OpenID identity is registered, the server application redirects the user to the login window. If the identity is not registered, the authentication process fails. The e-learning sever verifies the authorization status and allow or not to connect. If verification succeeded the user is now connected to the e-learning server. For example, the user created the OpenID URL: *mlearn.myopenid.com*. During the authentication process, the GET command includes the following parameters:

- `openid.identity` – value of OpenID identity (in our case: http://mlearn.myopenid.com)
- `openid.return_to` – the address of the redirected page after OpenID authentication

The advantages of using OpenID for e-learning Web based applications are:

- One account for multiple sites
- Open standard
- Many providers
- Easy implementation

There are some disadvantages also:

- Need implementation
- Desktop and mobile clients (non Web based) need extra work for implementation

The e-learning server is developed based on .Net technologies. The OpenID implementation is based on *DotNetOpenAuth*, an open library that implements OpenID 2.0 and 1.x and OAuth 1.0 and 1.0a. For Java- based servers, there is *openid4java* API [4].

6 Conclusions and Future Work

Using OpenID standard for Web-based e-learning applications increases the ease of use.
The forgotten user id or password should be less.
The user doesn't need to use and remember many credentials, one or two profiles being enough.
By encrypting the users' passwords will be ensured a high security level of the data inside the e-learning applications. Each user is responsible for the information registered in the application and he must protect his own authentication elements.
Future work includes implementation of this type of authentication on mobile devices. That includes Android and Windows Phone applications.

Acknowledgements

Part of this work was supported by CNCSIS – UEFISCSU, project number PNII – IDEI 2637/2008, project title: *Project management methodologies for the development of mobile applications in the educational system.*

References

[1]Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press
[2]Protocolul OpenId http://en.wikipedia.org/wiki/Openid
[3] OpenID Authentication 2.0 - Final http://openid.net/specs/openid-authentication-2_0.html
[4] OpenID for Java Web applications http://www.ibm.com/developerworks/java/library/j-openid2/?ca=drs-
[5] openid4java API http://code.google.com/p/openid4java/
[6] J. Erickson, *Beware Open Source Encryption*, http://www.drdobbs.com/open-

source/220800130
[7] F. Alecu, P. Pocatilu and S. Capisizu, WiMAX Security Issues in E-Learning Systems, *Proc. of 2nd International Conference on Security for IT & C in Journal of Information Technology and Communication Security*, Bucharest, November 2009, pp. 45-52.
[8] Rescorla, E., "Diffie-Hellman Key Agreement Method," RFC 2631.
[9] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174.