Single Sign-on with One Time Password

Paras Babu Tiwari
Software Engineer
D2Hawkeye Services
Bishalnagar, Kathmandu
Nepal.

Shashidhar Ram Joshi, Ph.D.
Professor and Head of Department
Department of Electronics and
Computer Engineering, Pulchowk
Lalitpur, Nepal.

Abstract

The organizations are shifting from the desktop based solution to web based platform to solve their business needs. The amass of these different systems poses problems for end-users, who must frequently provide their username/password across different systems, as well as for system administrators, who must manage security and access for those systems. Single sign-on mechanisms have become increasingly important in solving these problems. Although there are several ways to implement single sign-on feature (e.g. Kerberos and X.509), it may be difficult to modify a particular existing application to comply with the different protocols imposed by the new authentication mechanism. Moreover, these methods of implementing single sign-on require the new infrastructure which will cost both time and money. This might creates hindrance to the small organization for the implementation of single sign-on feature. Present study proposes a method of implementing single sign-on using the one time password. In this approach, when user manually authenticates in the portal site and tries to login into the subsequent applications then one time password is generated for that session and this password is used to authenticate the user into the applications. This method eliminates the necessity of setting up the new infrastructure and also the existing system requires minimal changes to incorporate the single sign-on feature in it. Hence, the organization can implement the solution in a cost effective manner.

I. Introduction

The widespread popularity of the web technology leads to the gradual shift of desktop based application to web based solution. The diverse set of the requirements of the user incurred the organization to develop the various web applications. For example consider a software company where an employee might need to use the applications for various operations like logging the daily activities, tracking the bugs in the system, managing the projects etc. Generally, the organization maintains separate applications for each distinct type of activities and when user tries to access any application then he/she is presented with the login screen. The need of user authentication in each application causes the trouble for the users as they need to remember the password for each system. In such situation, users have the tendency to write down the password in some notes so that they can remember it or use the same password across multiple applications. This increases the risk of cracking the password by intruder

In this situation, we can implement the single sign-on(SSO) feature to increase the user experience and make the system more secured. The single sign-on is a way to access the multiple, related, but independent software system in such a way that user logs in a system and gains the acess to all the system without being prompted to re-login in each application. There are several advantages of single sign-on.These are:

- It increases the productivity of the organization. The user is not mired by multiple logins and they don't require to remember username and password for each application and also the number of call about password help is reduced.
- It simplifies the IT adminstration by reducing the number of username/password that should be managed.
- It increases the security of the system.

There are several different and robust way to implement single sign-on feature. However, the implementation of these methods in the legacy systems requires the changes in the system and we need to build the new infrastructure to implement the different single sign-on methods provided by the different vendors. This creates a burden for the organization to implement the single sign-on. So there is a need of simple and cost-effective method of implementing the single sign-on feature in legacy systems.

In this paper, we presented a method of implementing single sign-on feature by combining the concept of one time password (OTP) with single sign-on. The OTP is a mechanism that prohibits the unauthorized access of protected resource like user account. The OTP approach entails the user to use different password for each login and it is widely used for two factor authentication. In our approach of implementing SSO, the OTP is used to establish the communication among the applications.

II. Related Work

Pashalidis, A., & Mitchell, C. J. [1] provides an overview of different methods of implementing Single Sign-On feature. The different types of single sign-on system categorized by authors are:

- Local pseudo-SSO systems.
- Proxy-based pseudo-SSO systems.
- Local true SSO systems.
- Proxy-based true SSO systems.

The popular SSO mechanism like Kerberos, Microsoft Passport etc fall in the proxy-based true SSO architecture. To implement these methods, there is a need of an external server which acts as an Authentication Service Provider (ASP). Moreover, these methods define set of protocols to establish the communication between users and Service Provider (SP) [1]. For example, the Kerberos system works by creating the short lived token based on the initial user authentication. To generate such token, the Kerberos system has predefined set of protocols and application participating in the single sign-on must implement these protocols. Moreover, the Kerberos needs authentication service, ticket granting server, and service server etc. to implement the protocol [2]. It is difficult to implement these varieties of protocols in the legacy application and also need of new infrastructure creates a burden to implement the solution for small organization.

An alternative and elegant solution in such situation is to use the password synchronization. In this approach, when user login into the first application then user's private data such as password is passed to the subsequent application through the network. This method is simple to implement in the existing system but it creates the major security risk by exposing user's private data in the network.

III. METHODOLOGY

In this section we will give the detail about the design and implementation of single sign on feature with one time password.

A. DESIGN

Fig.1 represents the flow of the system in which user login into the portal site where the user's credential is verified. We have maintained a database of the username and password and the credential is verified against this database. If the user inputs the parameter correctly then a welcome page is displayed that lists the applications on which user is authorized to access. On the other hand if the username/password does not match then appropriate message is displayed to the user.

1. The service providers are the applications that provide some kind of service to the user for e.g. inventory management system, operation and request management system etc.

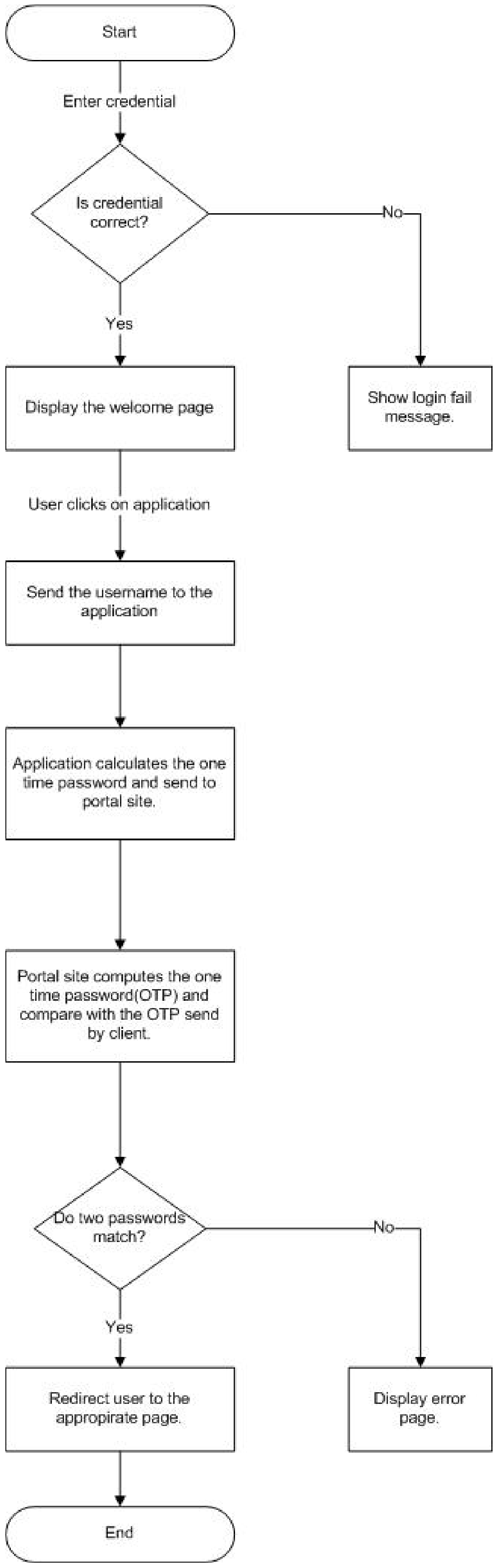
Start
Enter credential
Is credential correct?
No
Yes
Display the welcome page
Show login fail message.
User clicks on application
Send the username to the application
Application calculates the one time password and send to portal site.
Portal site computes the one time password(OTP) and compare with the OTP send by client.
Do two passwords match?
No
Yes
Redirect user to the appropirate page.
Display error page.
End

Fig. 1. Flow Diagram

Once user clicks on the link, we use the Leslie Lamport algorithm to generate one time password. The one time password generation algorithm depends on the concept of one way function. According to Lamport, let 'N' be the total number of time that one time password be generated. Consider a following one time function:

$$y=f(x). \tag{1}$$

The above function can be calculated in polynomial time in forward direction but calculating its inverse takes exponential time, and let 's' be a seed. We apply the function for 'N' time to generate the first password, N-1 time to generate the next password and so on. Let P_1 denotes the first password, and let N=3 then P_1 is given by

$$P_1 = \quad f(f(f(s))). \tag{2}$$

If any intruder is able to get the password at any time then it will be impossible to compute the next password in sequence because he/she needs to compute the inverse of the function. Since the function is chosen to be one-way, finding the inverse of the function is a hard problem [3].

Request For Comment (RFC 2289) provides a standard way to implement this function and our solution implements the standard mentioned in RFC to generate the one time password. RFC 2289 outlined the process of generating one time password as follows: "The OTP system generator passes the user's secret pass-phrase, along with a seed received from the server as part of the challenge, through multiple iterations of a secure hash function to produce a one-time password. After each successful authentication, the number of secure hash function iterations is reduced by one. Thus, a unique sequence of passwords is generated. The server verifies the one-time password received from the generator by computing the secure hash function once and comparing the result with the previously accepted one-time password [4]." In this implementation, md5 algorithm [5] is used as a secured hash function.

B. Technology

This project is implemented in the Java programming language. The project is web based and it uses the Java servlet technology along with Jsp pages [6]. The mysql is used as the backend and the Hibernate [7] is used to access the database through the application. We used the secure hash (SHA-1) message digest algorithm [8] to encrypt the password.

C. Implementation

To implement the solution, user's password is used as the pass-phase and for each user we store the challenge question in the database. The database is maintained by portal site only and it contains the username, password, total number of times the password should be generated and challenge questions for each user.

When user clicks on the link on welcome page of the portal site then username is sent to the application from this site. The application fetches the necessary data of the user from the database and computes the one time password for a particular sequence. After that, the password is encrypted by using the SHA-1 message digest algorithm and the application sends back the password to the portal site. The portal site again locally computes the one time password for the same sequence, encrypts the password by using same algorithm as used by client to encrypt password and compares the two passwords to determine whether they are equal or not. If both the password matches then authentication succeeds. The portal site decreases the total count by one in the database and user is redirected to the appropriate page. In the next login, the application first checks the value of the "total number of one time password" column to determine whether it is zero or not. If it is zero then user is forced to update the password and challenge questions. Once user enters the value, the database is updated with the new value of password, challenge questions and "total number of one time password" field is reset.

IV. Discussion

In this paper, an improvement over the password synchronization method is discussed. For this, the one time password mechanism is combined with the single sign-on feature. One time password is a popular mechanism of user authentication in which a new password is used in each login.

Our approach of implementing the single sign-on is particularly suitable for the organization which has to integrate the number of existing applications. Following are the advantages of our method of implementing the single sign-on.

- The execution of the true SSO like Kerberos is costly as we need to setup the infrastructure for its implementation [1]. On the other hand, there is no need of any extra hardware/software for the implementation of our solution. Therefore, it is less costly than the other method of implementing SSO.

- Kerberos SSO implementation requires a great change in the existing application to comply it with the Kerberos protocol [9]. But our solution can be implemented with the minimal change in the existing system. The major changes required to implement this solution is to write the program to generate the one time password. However, there is already a package named "java.security.MessageDigest" and MessageDigest class exposed number of methods which can be used to generate the one time password.

Moreover, If we expose the user's static password in the network then one can obtain the information by eavesdropping ,and the form of current password can be utilized to guess the future ones [10]. But our method of implementing SSO does not expose the user's static and long lived password directly in the network. By doing so, we have forestalled the chances of hacking the password by intruder. Similarly, it will be difficult for the eavesdropper to decode the password used for the communication between applications. There are two reasons for this:

- Complexity of the password.
- Variation of password.

Since the password is generated by using the algorithm, it is obviously more complex than the simple password used by the user for login. Beside this, the password varies in each login of the user and it is encrypted by using SHA-1 message digest algorithm which also increases the complexity in the process of decoding the password. If the password is decoded by any means then also interloper can't use it for the next time and he/she also can't compute the next password based on the previous password because of the one-way function used to generate the one time password.

V. CONLUSION and FUTURE WORK

We have discussed an improvement over the password synchronization method to implement the single sign-on feature. While there is several other robust method of implementing single sign on feature, they are generally infeasible when the organization wants to implement it in its legacy system with minimum changes. We have implemented a system in java which has minimum impacts on the existing system. There is no need of any extra hardware and software to implement this solution. Hence, the organization can afford the solution and implement it in their existing system. Similarly, we have used the one time password to communicate among the application which prevents the cracking of password by eavesdropping.

We have used md5 algorithm as a secured hash function to generate one time password but this algorithm also has the drawback. In the future, we will study the impact of the md5 vulnerability in our system.

REFERENCES

[1] A. Pashalidis and C. J. Mitchell, "A Taxonomy of Single Sign-On Systems" , in R. Safavi-Naini and J. Seberry (editors), Information Security and Privacy - 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11 2003, Proceedings, Springer-Verlag (LNCS 2727), Berlin (2003), pp.249-264.

[2] B.C. Newman, and T Ts'o,., "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications*, 32(9):33-38, Sept. 1994.

[3] Andrew S. Tanenebaum,. Modern Operating Systems, 2nd ed., India: Prientice Hall, 2004, pp.599-600.

[4] N. Haller, C. Metz, P. Nesser, and M. Straw, "A One-Time Password System ," *Internet Engineering Task Force Request For Comments 2289*, IETF Website, 1998. Available: http://www.ietf.org/rfc/rfc2289.txt

[5] R. Rivest, "The MD5 Message-Digest Algorithm", *Internet Engineering Task Force Request For Comments 1321*, IETF Website, 1992. Available: http://www.ietf.org/rfc/rfc1321.txt

[6] E. Armstrong, J. Ball, S. Bodoff,, D. B. Carson, I. Evans , D. Green, et al., The J2EE™ 1.4 Tutorial.. 8.2 ed., Santa Clara, CA: Sun Microsystems, 2005. [E-book] Available: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/J2EETutorial.pdf

[7] G. King, C. Bauer, M.R. Andersen, E. Bernard, E. Bernard, and S. Ebersole, Hibernate Reference Documentation, 3.3.2 GA, Raleigh North Carolina, USA: Red Hat, Inc., 2009. [E-book] Available: http://docs.jboss.org/hibernate/stable/core/reference/en/pdf/hibernate_reference.pdf

[8] National Institute of Standards and Technology,U.S. Department of Commerce. Secure Hash Standard, 2002. FIPS PUB180-2. Available: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[9] T. Fleury, J. Basney, and V. Welch, "Single sign-on for java web start applications using myproxy," in SWS '06: Proceedings of the 3rd ACM workshop on Secure web services. New York, NY, USA: ACM Press, 2006, pp. 95-102.Available: http://grid.ncsa.illinois.edu/papers/sws-myproxy-jws.pdf.

[10] N.M. Haller, "The S/Key One-Time Password System." *Proc. Internet Society Symposium on Network and Distributed System Security*, pp 151-157. San Diego, CA, February 1994. Available: http://www.cs.utk.edu/~dunigan/cns04/skey.pdf.