International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 2, February - 2014

# A Novel Approach To Enhance The Security Of Web 2.0 Mashup Technology

[1.]Kajal Zambare, [2.]Shital Bandal,[3.] Madhuri Aware, [4.]Prof. P. A. Bandgar
[1,2,3,4.] Department of Information Technology, BSIOTR (W), Wagholi, University of Pune, Pune, India.

***Abstract* --Web Mashup is the new way to create a web application, by integrating data and services from other various web applications and data sources. Several technologies like AJAX, RSS, ATOM, REST and XML etc. have emerged, which are used in creating mashups. While combining diverse content or services from hetrogenious sources into a new one, different security issues arises such as user privacy, data confidentiality, data integrity and user authentication.**

**This paper is aimed to provide a security that will use cryptographic techniques to address the issues of data confidentiality, data integrity and authentication. We also provide protection against the most common attacks like XSS and CSRF attacks in web mashups.**

***Keywords- Web 2.0, Mashup Security, Cryptography, Privacy, SOP.***

## I. INTRODUCTION

A number of new techniques for the creation of web applications have been resulted into the rapid growth of Web 2.0; one of these techniques is meshing up of required content/services from several independent sources for the purpose to create a new content/service.
The term "mashup" has been originated from the music field where it means that producing/composing a new song by mixing lyrics and background music from several different source songs.

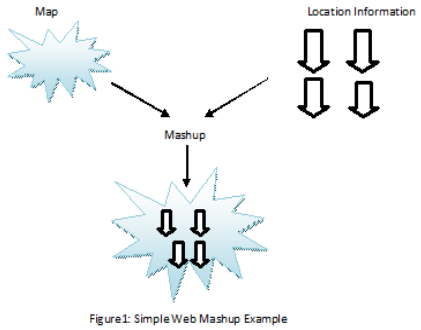

Figure1: Simple Web Mashup Example

In figure 1, we are integrating two different web applications, such as map and location information into new one. A web mashup can be created very quickly because it is based on the integration of information already provided by different sources on the web. A simple example can be of a web page extracting map information from one source and location information of a service (such as restaurants or houses for sale or rent) from another site and display the map with services placed on it as shown in Fig. 1. For example, HousingMaps.com integrates map information from Google Map and real estate information from Craigslist .com. A web site containing a web page that combines these resources is called integrator and the web site that provides the contents that integrator wants to use is called provider.

## 3. Web Mashup Security

Web mashups being primarily concerned with integrating content/services from different sources into a new web service, therefore introduce new security issues. Security issues can arise either from the security problems or loop-holes of the technology being used or the nature of the mashups to be performed.

The security model Same-Origin Policy implemented by browsers also creates problems for web mashups security. SOP uses origin to classify documents: documents belonging to the same origin can openly access each other's contents whereas documents from different origins are not allowed; origins are recognized
by hostname/lntemet domain, protocol along with port. SOP imposes some restrictions on scripts; violation of any of them may result in security breach :

1. One origin script should not be able to read or write content to/from another document or frame belonging to another origin.
2. One origin script should not be able to access the JavaScript environment of a frame belonging to another origin.
3. One origin script should not use XMLHttpRequest to communicate with a site belonging to another origin.
4. Scripts' access to cookies and browser plug-ins is restricted by the Same-Origin Policy.

## II. SECURITY ISSUES

Techniques used to create web mashups completely ignores the Same-Origin Policy and a web application violating the SOP would be a good working example of web mashup as an integrator would combine information from different content providers belonging to different trust

domains. Hence a new security framework is required to be built where issues like user privacy, data integrity, data confidentiality, and user authentication are properly addressed:

### *A. User Privacy*

Content providers allowing others to retrieve their information using their APIs might be wishing that their content should be available to a particular web site and not to any other unwanted parties. This will enable them to determine the unfair use of their content .

### *B. Data Integrity*

The user should be guaranteed that the content received by his web page is not corrupted by any untrusted party on the way before being received. In other words, the user will be receiving the contents as they are sent by the sending body. Similarly, the content provider or integrator will wish that other parties should not be able to corrupt the content being sent to the user for display.

### *C. User Authentication*

The communicating entities may wish to guarantee and verify the identities of each other to build trust measures.

### *D. Data Confidentiality*

The communicating entities may wish to guarantee that the content being exchanged should be readable to the intended parties only, any other third party on the way should not be able to disclose content of the packets being exchanged.

## III. PROPOSED SOLUTION

As two of the most common attacks on web mashups security are XSS and CSRF. XSS can be avoided by specifying session timeouts, regenerating session identifiers with each request or whenever privacy is essential the user should renter his credentials. CSRF can be avoided by enabling servers not to accept HTTP GET request, which is practically impossible; therefore, another way is: a token should be transmitted to the server with each HTTP POST and GET request. Most of the current research work basically focuses on providing solutions to XSS and CSRF; whereas, the issues of data integrity, data confidentiality, user privacy, and user authentication have not been addressed explicitly and are left for the web protocols or technologies used for sending, receiving, and storing of data. Therefore a composite framework is needed that will take into account all these issues. So, the use of cryptographic techniques can help us more in this regard.

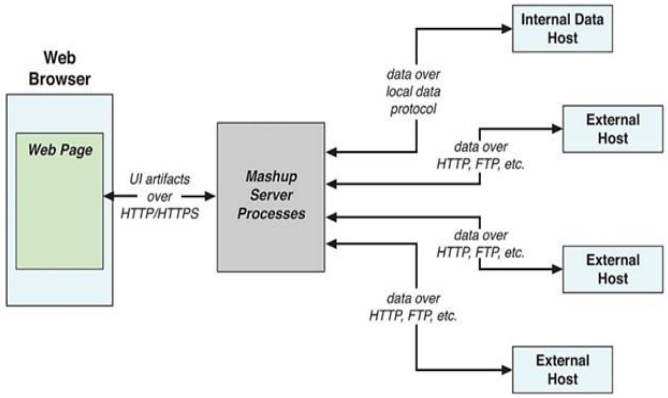


Fig.2 Server Side Web Mashup Model

The proposed solution considers Server-Side web mashup architecture, where the mashup components are combined on the server and users requests them for content providing as shown in Fig. 2. The proposed solution uses cryptographic techniques. This architecture consists of two phases: Phase 1 and Phase 2.

### *A. Phase 1*

Phase 1 consists of three steps (three-way handshake) taking place between a user and a web mashup server shown in Fig. 3.

1. In step I, user sends a message consisting of a request along with user 10, users' public key and a nonce ( a number generated with each session request uniquely and to provide guard against replay attack). The message has been encrypted using the public key of web mashup server, which is assumed to be available to the user.
2. In step 2, web mashup server responses with a message encrypted using users' public key, where the message contains request-specific array of tokens for the user, session timeout policy, session 10, a nonce which is unique with each user unique request, and a function applied to the nonce sent by the user showing that the user request message is received and correctly interpreted by the web mashup server.
3. In step 3, user responses with a message encrypted using web mashup servers' public key, where the message contains a function applied to the nonce send by the web mashup server showing that the server message is received and correctly interpreted by the user, and request specific array of tokens send by the mashup server for the user showing that user has agreed upon on the server selected request specific array of tokens for the user.

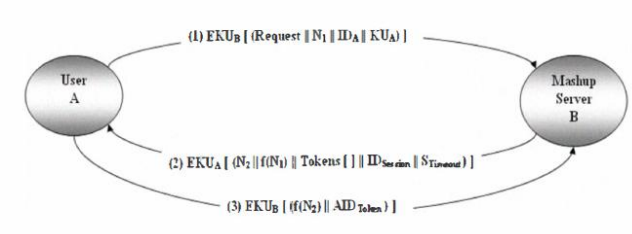


Figure 3: Exchanging of control information between user and web mashup server.

### *B. Phase 2*

Phase 2 consists of two scenarios slightly different from each other as shown in Fig. 4. Fig. 4 (a) shows transmission of user request to the web mashup server and Fig. 4 (b) shows web mashup server's response to the user's request.

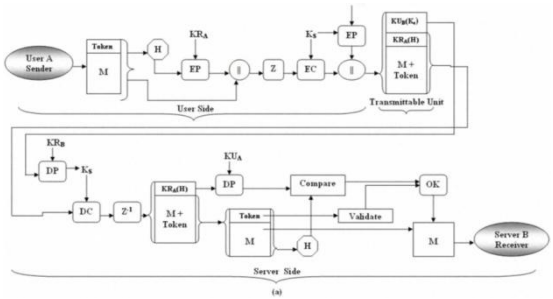


Fig.4 Phase 2(a) Request from user to web mashup server

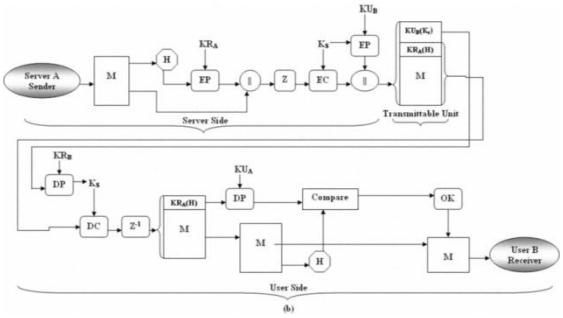


Fig.4 Phase 2(b) Response from web mashup server to user

**1) User Authentication and Data Confidentiality:**

User authentication and data conditionality is required for the protection of information to be communicated and is achieved in similar way in both of the scenarios.

**User Authentication:** On the sender side, the hash value calculated is encrypted using the private key of the sender (KRA), the encrypted hash value is decrypted using the public key of the sender (KU A) at the receiver side.

**Data Confidentiality:** On the sender side, sender selects a session key (Ks) and encrypt the concatenated
message using the selected session key. The session key is also encrypted using the public key of the receiver (KUB), and is combined with the encrypted concatenated message to be transmitted to the receiver. On the receiver side, the encrypted session key is first extracted and is decrypted using the private key (KRB) of the receiver. The encrypted concatenated message is then decrypted using the decrypted session key. Thus, an unauthorized user having no knowledge of the concerned keys if catches the message on transit would
not be able to decrypt it and disclose the message contents.

**2) Data Integrity:**

Again achieved in the similar way in both of the scenarios. On the sender side, SHA-I (Secure Hash Algorithm - 1 RFC 3174) is used to generate the hash code and the sender's private key (KRA) is used to encrypt the hash code and is concatenated with the message to be sent to the receiver. On the receiver side, the hash code is separated from the message and decrypted using the sender's public key (KUA). Hash code of the received message using the same SHA -I is
Calculated by the receiver. The received message will be considered accurate and accepted if the calculated hash code and the decrypted hash code are found to be the same, and will be rejected otherwise.

**3) CSRF Attack:**

If CSRF attack is initiated on the web mashup server then SOP, user name/password, cookies, and SSL will not provide protection against this attack. Therefore, a request specific token should be included within each HTTP GET and POST request to protect against CSRF attack . A request-specific token is associated with the message by the sender as shown in Fig. 4 (a), which is verified by the receiver and if found correct the message is accepted otherwise rejected. Response from the web mashup server to the user is not needed to use the token as shown in Fig. 4 (b).

**4) XSS Attack:**

As XSS attack is concerned with invalidating of the previous/existing session, the session time out policy information sent by the web mashup server in step 2 and accepted by the user in step 3 in phase 1 can be used to guard against the XSS attack.

## IV. CONCLUSION AND FUTURE WORK

Web application development has entered in a new era with the introduction of Web 2.0. Instead of creating web application form scratch, relevant content can be extracted from online available sources and meshed together and presented in a unique way not present earlier. This new approach to software development can pose many security challenges that bypass the domain of cross site referencing and issues in data integrity, user authentication, and data confidentiality emerge. The inherent security model (Same-Origin Policy) implemented in web browsers as well as cookies, and SSL etc cannot deal with these security challenges and hence necessitates a new security framework. This research paper presents a security framework using well-known cryptographic techniques that can be used in Server-Side mashup model and will provide solutions to most common mashup security attacks such as CSRF, XSS and other relevant security issues.

However, the proposed model has the capacity to be extended. There are certain other parameters such as access control and non-repudiation which are also essential for a system's security. These aspects are not covered in the current proposed solution and are left for the future extension of the framework.

## REFERENCES

[1] A Cryptography-Based Approach to Web Mashup Security Shaukat Ali, Shah Khusro, Azhar Rauf Department of Computer Science, University of Peshawar,Peshawar,Pakistanhoonikhan@mail.com,khusro@upesh.edu.pk, azhar_rauf@upesh.edu.pk

[2] W.Wendy(2010 March 02), Web Mashup – A new breed of web application[online],Available:http://www.wiliamcomaulwiam-bloglweh-mashup-a-new-breed-of-web application,[23/03/20II].

[3] A Bohannon. (2008), Building Secure Web Mashups [Online], Available: File: mashups.pdf,[11/03/2011].

[4] F.Thomas, An Overview Of Current Approaches to Mashup Generation, In Wissensmanagement-WM,vol:145,pages 254-259, Gi ,2009.

[5] J. J. Hanson (2009), Mashup Security[online], Available http://www.ibmcom/developerworkslwebllibrary/wamashup securel File: wa-mashupsecure-pdf.pdf,[11/03/2011]

[6] J. Hakkola. (2010), Mashup Security [Online], Available: http://www.tml.tkk.fi/OpinnotlT-111.5550/2008/ file: seminaai_hakk0lajyrki.pdf,[14/03/2011]

[7] W. Stallings, "Cryptography and Network Security Principles and Practices", Prentice Hall, 4th Edition, ISBN 0132023229, 2006

[8] D. Eastlake (2001), Secure Hash Algorithm 1 [Online], RFC 3174, Available: http://www.rfc.editor.orglrfc/pdfrfc/ File: rfc3174.txt.pdf.[17/03/2011]

[9] WikiPedia, Mashup (Web Application Hybrid) [Online],Available: http://en.wikipedia.orglwikiMashup_(web_application_hybrid), [15-03-2010]