Using Logically Hierarchical Meta Web Services to Support Accountability in Mashup Services

Ali Khalili , Shahriar Mohammadi
IT Group - Faculty of Industrial Engineering
K. N. Toosi University of Technology, Tehran, Iran
ali@ajaxian.ir , smohammadi40@yahoo.com

Abstract

Mashups are next generation of web applications; they integrate and remix different sources on the web in a creative approach to provide rich and novel experiences for users. Furthermore, mashups introduce a new class of integration technologies for implementing Situational Applications (i.e. applications that come together for solving some immediate business problems). While mashup services provide flexibility and speed in delivering new valuable services to consumers, the issue of accountability associated with the mashups remains largely ignored by the industry. Pushing mashups to enterprises without attention to accountability problems involves many risks. In this paper, a new model is proposed to resolve accountability issues in mashup services. The proposed model uses PKI (Public Key Infrastructure) in conjunction with Logically Hierarchical Meta Web Services to support identification and traceability of web services in mashups and consequently provides a trusted environment for enterprise mashups.

1. Introduction

Web 2.0 is becoming popular among people who are interested in creating or providing more useful services on the Internet [8]. The promise of remixing existing online services and data into entirely new online applications in a rapid, inexpensive manner, often referred to as **mashups**, has captured the software industry's imagination since the release of first major example, HousingMaps.com, in early 2005. Since then, mashups have offered the potential to finally make widespread software reuse a reality, enable SOA (Service Oriented Architecture) initiatives to achieve positive ROI, and radically drive down the cost of application development while satisfying large applications backlogs that plague organizations almost everywhere [10]. Mashups establish an emerging trend called "Situational Applications" where applications are constructed "on the fly" for some transient need.

The core Web 2.0 principles are "simple, low-barrier and fast" and "every user himself is the center on internet". For meeting these goals, mashups allow consumers to draw upon content retrieved from external data sources (web services, data store, and web application) to create entirely new and innovative services [11].

While mashup services bring flexibility and speed in delivering new valuable services to consumers, the legal implications of using this technology are significant. Researchers in law conclude that the development of mashup web services is fraught with potential legal liabilities that require careful consideration [16].

The issue of accountability associated with the mashup practice remains largely ignored by the industry. Current formal practices suggest that the mashup developer and original content source owner disclaim any warranties. This appears to be temporarily acceptable since most services from Web 2.0 sites are free to internet users. This means that as long as consumers accept the terms and conditions, the issue of accountability is largely avoided. Notwithstanding, as these services mature to be used in enterprises or involve some payment, such an approach may no longer be tenable to all parties [2].

Accountability in mashup services includes authentication of all the parties involved and traceability in service composition. In this paper we aim to solve the former by using PKI (Public Key Infrastructure) and assigning a digital certificate to each web service involved in mashup service; and then by creating a "Hierarchical Meta Web Services" to track execution of nested web services, solve the latter. A Meta Web Service is a third-party web service that is responsible for monitoring workflow of a mashup service without affecting internal operations of it.

The rest of the paper is organized as following. In section 2, we are going to present a background about mashup services, PKI and JSON. In section 3, we analyze accountability implications in mashup services. In section 4, we describe our proposed model for accountable mashup services that is based on PKI

978-0-7695-3473-2/08 $25.00 © 2008 IEEE
DOI 10.1109/APSCC.2008.55

and hierarchical Meta Web Services. Finally we conclude in section 5 and express the future works.

2. Background

2.1. Overview of Web Mashups

The term mashup (also mash up and mash-up) stems from pop music (notably hip-hop); it refers to practice of producing a new song by mixing two or more existing pieces. In the context of Internet, a mashup is a web application that combines data from more than one source into a single integrated tool [1]. Based on the concept of service composition in Service Oriented Architecture (SOA), mashup provides flexible and dynamic services with rich experience [2].

Mashups have recently emerged as a powerful applications development platform that combines multiple sets of data streams into a unified user experience [13]. The ProgrammableWeb.com has been an important resource in charting the development of mashups. According to the ProgrammableWeb, the number of mashups exceeds 3000, and is growing steadily at the rate of approximately 3 mashups a day. Most of the current mashups are ad-hoc, non-commercial experiments, built by users as an entertainment (Figure 1). Nevertheless, with maturing the technology, it is predicted that mashups and composite applications will be the dominant model (80%) for creating composite enterprise applications by 2010 [14]. Forrester [15] projects that the enterprise mashup market will reach nearly $700 million by 2013 and software vendors with mashup platforms will be ready to grab "the lion's share" of the market.

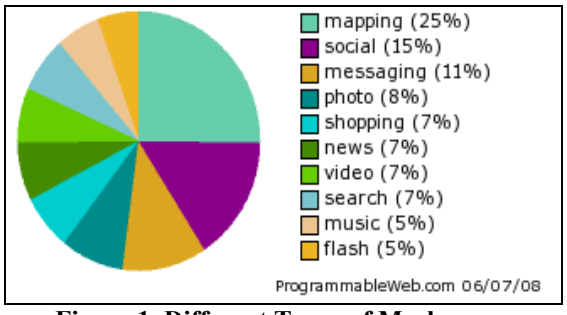

Figure 1. Different Types of Mashups

Security is a big concern in mashup services. Because mashups bring together content from multiple sources, they must somehow circumvent the traditional same origin web security model to obtain third-party data. Often web developers are forced to choose between security and functionality [12]. There are many proposed methods to address the security problems of cross-domain communications in mashup services. These proposals can be largely divided into three groups: proposals working with unmodified browsers; extensions to HTML requiring browser modifications; solutions based on browser plugins [5].

Two prominent methods that work with unmodified browsers are Subspace [12] and SMash [5]. One of the deficiencies of Subspace is that it requires complete trust of the component providers in the mashup provider as their code is executed in DNS domains controlled by the mashup provider [5]. In this paper, we choose to use SMash because of its compatibility and comprehensiveness in comparison to other approaches. In addition it uses a message-passing interaction style instead of a shared-memory style and is appropriate for our purpose.

2.2. Public Key Infrastructure (PKI)

A Public-key infrastructure (PKI) is a system for publishing the public-key values used in public-key cryptography. A PKI integrates digital certificates, public-key cryptography, and certificate authorities into total, enterprise-wide network security architecture [7]. In this paper we use PKI for two purposes:

1. Identifying the parties involved in mashup service
2. Internal communications of parties involved in mashup service.

2.3. JavaScript Object Notation (JSON)

JSON (JavaScript Object Notation) [6] is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language. JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

In this paper, we choose to use JSON format for specifying orchestration and choreography of mashup services because it is a lightweight format in

comparison to standard xml-based formats like WSBPEL [9] and can be evaluated quickly in client browser.

3. Accountability in Mashup Services

The meaning of the term accountability appears to vary considerably and is dependent upon the context. Traditionally the topic of accountability has attracted much interest with focus on the e-commerce transaction. According to Kailar, accountability is "the property whereby the association of a unique originator with an object or action can be proved to a third party". The definition implies non-repudiation in an e-commerce transaction [2]. In [3], accountability is defined in multihop message communications that is similar to communication in mashups environment. It considers accountability as non-repudiation of following cases:

- Non-repudiation of Origin
- Non-repudiation of Receipt
- Non-repudiation of Submission
- Non-repudiation of Delivery

In [2], an extensive research on the concept of accountability and its meaning for mashup services is performed and accountability for multiple parties is defined, see Table 1 below.

Table 1. Accountability for Multiple Parties

Accountability in services refers to the obligation that several persons, groups, or organizations assume for the execution and fulfillment of a service. This obligation includes:
• Answering, providing an explanation or justification, for the execution of that authority and/or fulfillment of that responsibility;
• Full disclosure on the results of that execution and/or fulfillment;
• Undeniable liability for those result (non-repudiation); and
• Obtain trusted agreement of accountability from all entities involved in the service. Who in turn are bound to the obligations set out above.

According to these definitions, we focus on two aspects of accountability:

1. Authentication: Identifying all the parties involved in a mashup service.
2. Non-Repudiation: traceability in composition of mashup services and monitoring internal interactions.

Furthermore, we consider security solutions for cross-domain mashup communications that is necessary for implementing our proposed model.

4. The Proposed Model

Our proposed model as shown in Figure 2 uses a third-party web service called Meta Web Service (MWS) as well as a Communication Bus for each mashup service. Before describing these components, we should consider the following assumptions:

- A digital certificate is assigned to each web service involved in mashup service for authenticating identity of it.
- Meta Web Services rely on a commonly accepted agreement between mashup service providers. MWS uses a secure and robust infrastructure that is considered trustful for all mashup services.
- Mashup services use a commonly accepted assembly model and have a regular communication model.

Considering these assumptions, each mashup service that want to be accountable must add a MWS to its communication bus.

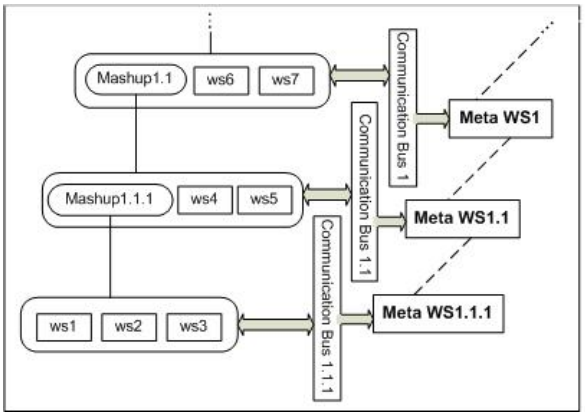

Figure 2. The Proposed Model

4.1. Meta Web Services (MWS)

A Meta Web Service is a web service that includes information about another web services, it has some procedures to track actions of another web service. In the context of mashups, a MWS is a third-party web service that is responsible for monitoring workflow of a mashup service without affecting internal operations of it. Each MWS receives the following information about a mashup service:

- **Orchestration**: Decomposes capability of mashup in terms of the functionality required from other services. An orchestration as expressed in [4],

must necessarily define three aspects: (1) the participants in the composition; (2) the control-flow governing the order and choice between executions of these participants (3) the data-flow governing what is communicated between these participants. In our proposed model we consider a Communication Bus as expressed in [5] for coordination of services in mashups (Figure 3).Web services are isolated from each other and can only communicate with each other through the mediated channels. The communication bus is a publish/subscribe system with many to-many channels on which messages are published and distributed. Permission of each web service to read and write on each channel is clearly defined in this architecture. For example in Figure 3, web service A can publish to Channel 1 and Channel 3 and is subscribed to Channel 2 and Channel 3, Meta Web Service is a read-only web service that is subscribed to all channels.

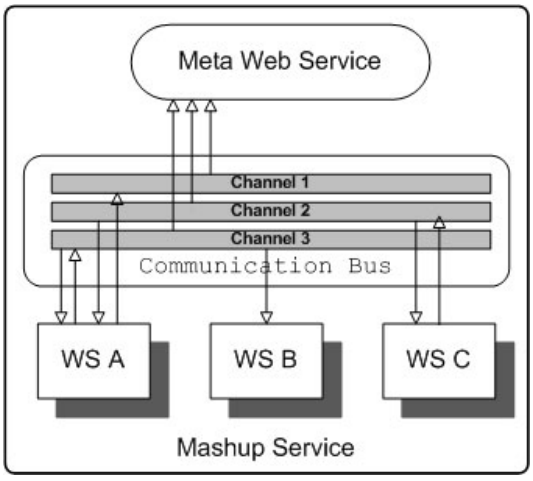

Figure 3. Mashup Communication Bus

We use a simple JSON [6] format for specifying orchestration of a mashup service (Table 2). This format is similar to WS-BPEL [9] standards. In this format, "partnerLinks" define the different parties involved in the process, "variables" define the data variables used by the process, "invoke" indicate invoking some Web service's operation, "sequence" is used for defining a sequential execution order, and "flow" is used for parallel execution.

Table 2. An Example for Specifying Orchestration of Mashup Services using JSON format

```
{
 "orchestration":{
  "partnerLinks":{
   "partnerLink":[
    {"name": "WS A", "pubicKey": "pka"},
    {"name": "WS B", "pubicKey": "pkb"},
    {"name": "WS C", "pubicKey": "pkc"}
   ]
  }
  "variables":{
   "variable":[
    {"name": "v1","type": "integer"},
    {"name": "v2","type": "string"}
   ]
  }
  "sequence":{
   "invoke":[
    {"partnerLink": "WS A","operation": "opA1",
    "inputVariable": "v1","outputVariable": "v2"},
    {"partnerLink": "WS B","operation": "opB2",
    "inputVariable": "v2","outputVariable": "v2"},
    {"partnerLink": "WS C","operation": "opC1",
    "inputVariable": "v2","outputVariable": "v1"}
   ]
  }
 }
}
```

- **Choreography:** Decomposes capability of mashup in terms of interaction with the mashup service. The choreography defines how to communicate with the mashup service in order to consume its functionality. As described in Orchestration, we can define a simple JSON format similar to WS-CDL [17] for specifying choreography of a mashup service. Since it is beyond the scope of this paper, we ignore describing the details.
- **Other Managerial Information:** Includes nonfunctional properties such as version information, accuracy (the error rate generated by the mashup service), financial (the cost-related and charging-related properties of a mashup service), owner (the person or organization to which the mashup service belongs), etc.

4.1.1. Logically Hierarchical Meta Web Services

To support accountability in mashup services, we need a hierarchical infrastructure for Meta Web Services. In fact there is only one MWS provider, however since it uses a hierarchical data model for sharing data between various instances of itself, we can view our model as logically hierarchical meta web services. Each MWS have information about involved parties in related mashup service that can be atomic web services or mashups that have their own MWS.

Meta Web Services are created in a bottom-up manner so each MWS receives information about its sub MWSs. We can demonstrate these relations using a directed acyclic graph (Figure 4).

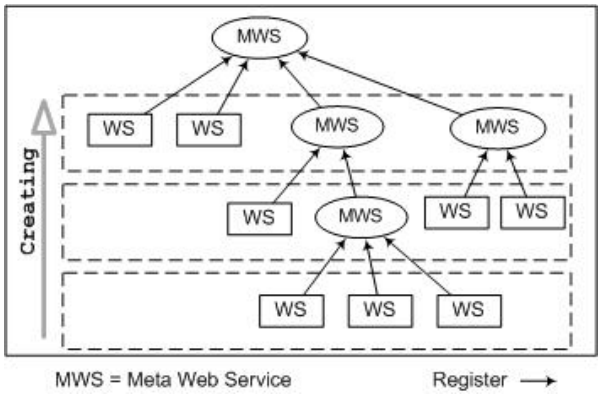

Figure 4. Bottom-Up Creation of Meta Web Services

While registering a new MWS for a mashup service we may encounter exceptional states that result in creating cycle in tree (e.g. Figure 5). We can easily prevent these situations by checking the condition of occurring them before registration of new MWS.

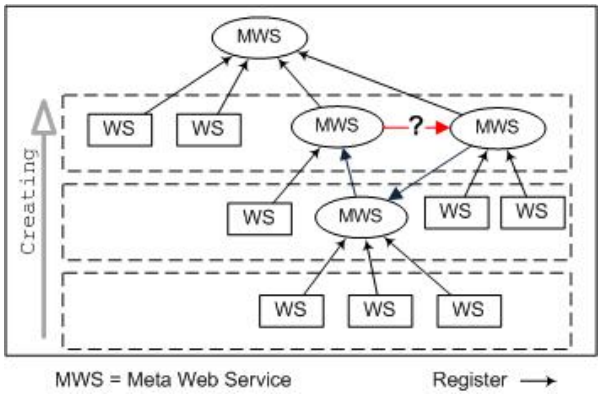

Figure 5. Exceptional State in Creating New Meta Web Service

4.2. Accountability Support

Logically Hierarchical Meta Web Services in conjunction with PKI provide a robust infrastructure to address accountability issues in mashup services. Each MWS have information about internal interactions within its related mashup and DFS (Depth First Search) algorithm can be used to trace execution of a mashup service (Figure 6).

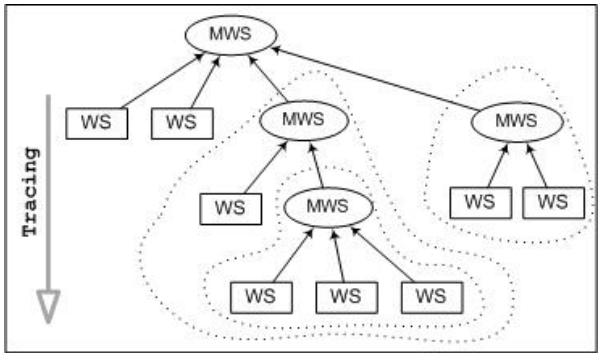

Figure 6. Tracing Execution of a Mashup Service

DFS algorithm as shown in Table 3, traces execution of a mashup service and can determine the origin of violation if there was any problem in system.

Table 3. Traceability by Using Meta Web Services

```
//input: a meta web service
procedure traceMashup(MWS s){
 Create a Queue Q based on s Service Orchestration
 While Q is not empty do
  for each web service v in Q do // involved services
   for all web services e that have relations with v do
    if relation e is unexplored then
      let w be the related web service of e.
       if web service w is unexpected then
        if type(w) is meta web service
           Recursively call traceMashup(w)
        mark e as a discovery relation
        //trace action
    else
      mark e as a back relation
}
```

5. Conclusion

In this paper, a new model for supporting accountability in mashup services is proposed. The proposed model uses PKI and Logically Hierarchical Meta Web Services to support identification and traceability of web services in mashups. Each mashup services that want to be accountable must add a MWS to its communication bus. A MWS is a third-party web service that is responsible for monitoring workflow of a mashup service without affecting internal operations of it. In this approach mashup services that are generated in bottom-up manner can be traced top-down using their Meta web service information.

Solving accountability issues in mashup services opens new doors for enabling enterprise mashups and lubricate rising situational applications. We envisage further evaluation of our model by implementing several instances of MWS in a real world mashup project and testing usability of our model.

6. References

[1] Mashup (web application hybrid), http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)

[2] J. Zou and C.J. Pavlovski, "Towards accountable enterprise mashup services", IEEE International Conference on e-Business Engineering (ICEBE 2007), 24-26 Oct. 2007, pp. 205-212

[3] S. Bhattacharya, R. Paul, "Accountability issues in multihop message communication", Proceedings of IEEE Symposium on Application-Specific Systems and Software Engineering and Technology, March 1999, pp.74–81.

[4] D. Fensel, H. Lausen, J. de Bruijn, M.Stollberg, D. Roman, A. Polleres ,and J. Domingue, *Enabling Semantic Web Services -The Web Service Modeling Ontology*, Springer Berlin Heidelberg, Germany, 2007.

[5] F. De Keukelaere, S. Bhola, M. Steiner, S. Chari,and S. Yoshihama, "SMash: Secure Component Model for Cross-Domain Mashups on Unmodified Browsers", International World Wide Web Conference Committee (IW3C2), 21–25 April 2008

[6] JSON (JavaScript Object Notation), http://www.json.org/

[7] W. Kou, *Payment technologies for E-commerce*, Springer-Verlag , New York, 2003

[8] Y. Nakano, Y. Yamato, M. Takemoto,and H. Sunaga, "Method of creating web services from web applications", IEEE International Conference on Service-Oriented Computing and Applications(SOCA'07), 2007

[9] Web Services Business Process Execution Language Version 2.0, 11 April 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[10] The 10 top challenges facing enterprise mashups, October 16th, 2007, http://blogs.zdnet.com/Hinchcliffe/?p=141

[11] X. Liu,Y. Hui, W. Sun,and H. Liang, "Towards Service Composition Based on Mashup", IEEE Congress on Services, July 2007, pp. 332-339

[12] C. Jackson, and H. Wang, "Subspace: Secure cross-domain communication for web mashups", 16th International Conference on the World-Wide Web, 2007, pp. 5-10

[13] N. Kulathuramaiyer, "Mashups: Emerging Application Development Paradigm for a Digital Journal", Journal of Universal Computer Science, 2007, Vol. 13(4), pp. 531–542

[14] Gartner's top 10 strategic technologies for 2008, October 09, 2007, http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9041738

[15] G. O. Young, "The Mashup Opportunity-How To Make Money In The Evolving Mashup Ecosystem", May 6, 2008, http://www.forrester.com/Research/Document/Excerpt/0,7211,44213,00.html

[16] R.S. Gerber, "Mixing It up on the Web: Legal Issues Arising from Internet Mashup", Intellectual Property & Technology Law Journal, Apsen Publishers, Aug. 2006, Vol. 18(8)

[17] Web Services Choreography Description Language Version 1.0, 17 December 2004, http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/