Available online at www.sciencedirect.com
ScienceDirect
journal homepage: www.elsevier.com/locate/cose

Computers & Security

A novel methodology towards a trusted environment in mashup web applications

Ahmed Patel [a,b], Samaher Al-Janabi [c,*], Ibrahim AlShourbaji [d], Jens Pedersen [e]

[a] School of Computer Science, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor Darul Ehsan, Malaysia
[b] School of Computing and Information Systems, Faculty of Science, Engineering and Computing, Kingston University, Kingston upon Thames KT1 2EE, United Kingdom
[c] Department of Information Networks, Faculty of Information Technology, University of Babylon, Babylon 00964, Iraq
[d] Computer Network Department, Computer Science and Information System College, Jazan University, Jazan 82822-6649, Saudi Arabia
[e] Department of Electronic Systems, Aalborg University, Aalborg, Denmark

ARTICLE INFO

Article history:
Received 28 April 2014
Received in revised form
27 September 2014
Accepted 17 October 2014
Available online 28 October 2014

Keywords:
Application programming interfaces (API)
Mashup applications
Risk analysis
Security
Threats
Risk filtering data mining algorithm (RFDM)
Confusion matrix

ABSTRACT

A mashup is a web-based application developed through aggregation of data from different public external or internal sources (including trusted and untrusted). Mashup introduces an open environment that is exposed to many security vulnerabilities, threats and risks. These weaknesses will bring security to the forefront when developing mashup applications and will require new ways of identifying and managing said risks. The primary goal of this paper is to present a client side mashup security framework to ensure that the sources for mashup applications are tested and secured against malicious intrusions. This framework is based on risk analysis and mashup source classification that will examine, analyze and evaluate the data transitions between the server-side and the client-side. Risk filtering using data mining suggests a new data mining technique also be utilized to enhance the quality of the risk analysis by removing most of the false risks. This approach is called the Risk Filtering Data Mining algorithm (RFDM). The RFDM framework deals with three types of clusters (trusted, untrusted and hesitation or unknown) to handle the hesitation clusters. Our proposal is to employ Atanassov's Instuitionistic Fuzzy Sets (A-IFs) as it improves the results of an URL. Finally, the results would be evaluated based on five experimental measures generated by a confusion matrix, namely: Accuracy (AC), recall or true positive rate (TP), precision (P), F-measure (considers both precision and recall) and F_β.

* Corresponding author.
E-mail addresses: whinchat2010@gmail.com (A. Patel), samaher@itnet.uobabylon.edu.iq (S. Al-Janabi), i_shurbaji@yahoo.com (I. AlShourbaji), jens@es.aau.dk (J. Pedersen).

http://dx.doi.org/10.1016/j.cose.2014.10.009
0167-4048/© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Mashup is an exciting interactive web application that draws upon diverse content retrieved from external data sources to create entirely new and innovative meta-application services (Na et al., 2010). Although mashup implies “unstructured” by definition, it is organized, serving as a weaver that systematically aggregates and stitches together third-party data. For example, one could combine online weather data with a virtual web-based map by integrating a geospatially-indexed temperature feed with a Google Maps interface. This leads to creating a new service that is neither supplied by the provider of the geospatially-indexed temperature feed nor by Google Maps service.

The main characteristic of mashup is the aggregation of contents or program codes from various sources into one integrated webpage to be displayed on the client side (user browser); these sources could be external or internal, trusted or untrusted sources (Yu et al., 2008; Merrill, 2009; Bianchini et al., 2010). Mashup is important to optimize the use of existing data, and is also flexible to obtain the maximum benefits from its personal and professional use. In the past few years, more and more web applications' providers have published Application Programming Interfaces (APIs) that enable software developers to easily integrate data and functions instead of building applications by themselves (Na et al., 2010).

In most cases, mashups lack the ability to gather and integrate different services in a secure way as the services may have completely diverse security requirements in terms of authentication and authorization (Meng and Chen, 2009). As a consequence, the tools that are used to construct mashups have been focused only on integrating security-free data sources and omitting other sources that do not have proper certification for publishing APIs (Yu et al., 2008).

The degree to which the Internet users feel that a web site protects their privacy may also have an impact on their trust of the site. Although perceptions of security and privacy protection are likely to differ according to each user's, traditions, culture, social networking and other factors, web site design and content elements can exchange privacy credentials to ensure user information is protected. With regard to design, site complexity and layout for presence and number of hyperlinks, obtrusiveness of advertising, ease of navigation, and general professionalism have been shown to affect perceptions about the authority of the site (Na et al., 2010). Perceptions on the site's authority may impact the user's privacy perception of the site. Web site content also influences perceptions of a site's privacy protection through privacy credentials, certificates and seals provided by trusted third parties, such as TRUSTe (TRUSTe, 2014), BBB online (BBB, 2014), or VeriSign (Verisign, 2013).

Clearly this can be a problem in enterprise environments because users are very reluctant to reveal their authentication information to third parties. Moreover, many mashups are built by non-technical users with no absolute guarantees that they will not accidentally leak important private or confidential data (Zou and Pavlovski, 2007; Rosenberg et al., 2009). As more new APIs and data sources become available to make mashup, system, vulnerabilities and security risks also increases. In order to continue such open integration it is essential and highly recommended that the integrated data are trusted and secured against malicious activities (Jackson and Wang., 2007).

The main goal of this work is to propose a mashup security framework that examines, analyzes and evaluates data transition between the server and the client-side (in client-server architecture), to ensure that the data sources are secure against security threats and malicious activities. This framework will implement a multilevel protection mechanism, which classifies the data source into trusted and untrusted source origins. The classification is based on an offline risk analysis process and an online monitoring of the data exchanges between the API providers and the client browser by measuring the residual risks and the sensitivity of the areas and assets that are required to be accessed. The proposed security framework will block/disallow the execution of mashup programs that display high risks to the client side during runtime operation.

2. Client-side mashup architecture

Mashups are web applications that integrate multiple data sources or API's into one integrated interface (Yu et al., 2008). Mashups typically allow the end user to discover and integrate a third party Ajax-powered mashup component onto a web-based application or website. Mashups can be considered to have an active role in the evolution of social software, Web 2.0 and Web 3.0 technologies (Bianchini et al., 2010). In a client-side mashup, the service or content integration takes place on the client side, which is typically a web browser. This is in contrast to a server-side mashup, where the service or content integration takes place in the server. A server-side mashup is also called a *proxy-style mashup* because a component in the server acts as a proxy to the service (Hilton, 2009). As illustrated in Fig. 1, a typical mashup framework is comprised primarily of:

a) End user Browser: The end user or client browser, where the data and application program code will be mashed-up and displayed. It is the space where the integrated application programs and scripts will be executed and processed at runtime.
b) Mashup Provider: The website where the mashups and required JavaScript libraries will be hosted. The products functionality can be accessed using the API services.
c) Data: The core element of any mashup is the data being aggregated and presented to the user; the data can strictly come from web services where data is serialized to Extensible Markup Language (XML) or JavaScript Object Notation (Barth and Li, 2011).

The primary reason for using the proxy style is to contend with the basic security protection that thebrowser security sandbox provides. In a proxy-style mashup, a server-side proxy allows access to the service without the view of the browser security sandbox, thus permitting connection to a site other than the server of origin to access a service.