SandMash: An Approach for Mashups Techniques on Smartphones

Raed Ali and Kalman Graffi

Technology of Social Networks Group, University of Düsseldorf, Germany
Email: ali_raed@cs.uni-duesseldorf.de, graffi@cs.uni-duesseldorf.de
WWW home page: http://tsn.hhu.de

Abstract. Supporting Mashup on mobile devices allows supporting advanced use cases and thus to accelerate the creation and combination of smart mobile applications. In this paper, we evaluate the three client-side Mashups proposals JS.JS, OMash and SMash on mobile devices. Our evaluation on mobile devices shows that the SMash proposal by IBM is reasonably suited for mobile mashups development as it requires less amount of effort from developers and at the same time it has cross-mobile-browser compatibility. In order to address the security, we integrated a sandbox functionality. We have modified the OpenAjax JavaScript library proposed in SMash and have added support of HTML5 ⟨iframe⟩ tag's "sandbox" attribute to it. ⟨iframe⟩ "sandbox" attribute, mobile mashups developers can restrict the framed-content (which may not be trustworthy) in a low-privileged environment. We demonstrate our proposal on a mobile mashup application that integrates content from three different providers (i.e., News, Stock and Weather service).

Keywords: mashup, smartphones, mobile services

1 Introduction

According to [8] mobile Internet and mobile devices will be available to 2 to 3 billion people over the next decade taking a huge impact on the media industry [7]. Nowadays, mobile phones have HTML and JavaScript compliant browsers. This makes mobile devices a potential and future venue for mashups like functionality. Mashups are web applications that combine data from different content providers at a single place i.e., web browser. The goal of mashups is to provide rich and better browsing experience to the users by taking advantage of client-side AJAX (Asynchronous JavaScript and XML). Third-party contents normally comes in the form of JavaScript code which may or may not be trustworthy. Third-party content, if malicious, may cause harm to the web application and its users. Researchers have proposed several solutions such as, JS.JS [14], OMash [2] and SMash [5], to confine third-party scripts. The motivation for this work is to evaluate client-side, web-based mashup solutions on mobiles which are nowadays essential part of our daily life.

In an earlier work [12], Philippe De Ryck et al. have identified the following Mashups requirements for building secure mashups: Separation, Interaction and



Fig. 1. Mobile Mashup Application

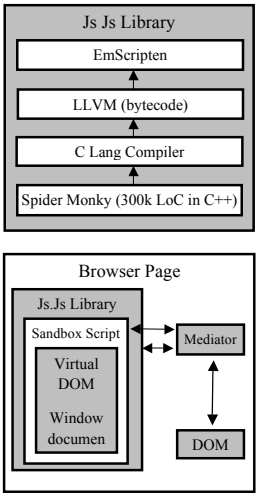


Fig. 2. JS.JS Architecture

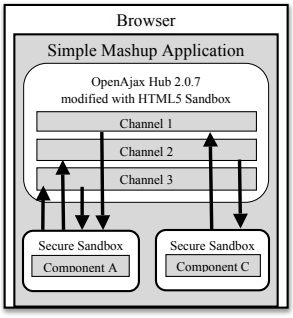


Fig. 3. Extended SMash Architecture with Sandbox Support

Communication. *Separation* requires that the DOM tree of one mashup component should be completly separated from DOM tree of other mashup components. At the same time scripts belong to individual component and should not be regulated by other components' scripts. In the literature, we found several techniques proposed by researchers for "separation" requirement. The proposals listed in this umbrella are: JS.JS [14], HTML5 ⟨iframe⟩ sandbox attribute [13], Google Caja , AdSafe and AdSentry [3]. *Interaction* should be secured. In mashup applications, content from different providers often need to interact with other party's content and with the provider application. During the interaction the confidentiality and integrity of private information should remain intact. We refer to [12] for detailed discussions on these secure mashups requirements. We found the following techniques in this category: OMash [2], OMOS [16] and Post Message. Mashups should support same-origin and cross-origin *Communication*. SMash [5] and W3C CORS7 proposals fulfill this mashup requirement

In this paper, we have evaluated one client-side mashup technique from a set of requirements on mobiles. The main contribution of this paper is to evaluate Mashups technologies (i.e., SMash, OMash and JS.JS) on mobiles, see for that Section 2. In Section 3, we revisit SMash's implementation (i.e., OpenAjax JavaScript library) and add a support for ⟨iframe⟩ "sandbox" attribute. To the best of our knowledge, this paper presents the first evaluation of web-based mashups techniques on mobiles.

2 Evaluation of Web-based Mashups for Mobiles

In this section, we discuss our evaluation of web-based mashups on mobile devices. During mashups evaluation on mobiles, we consider the following factors: 1) lightweight mobile applications, 2) cross-mobile-browser support, 3) amount

of effort required from developers and 4) the developers' learning curve. Considering this, we evaluate SMash, OMash and JS.JS on mobiles using the following running example, which simulates the real-life mashup application. In our mobile mashup example, content and data is coming from three different service providers i.e., stock news, weather information and news information. Fig. 1 shows our mashup application that we used to evaluate the mashup techniques.

2.1 Evaluation of JavaScript in JavaScript (JS.JS)

Terrace et al. proposed in [14] a client-side mashup solution named JavaScript in JavaScript (JS.JS). The main goal of JS.JS is to sandox third-party scripts which may or may not be trustworthy. Working According to [4]: JS.JS is a JavaScript interpreter running in JavaScript that allows an application to execute a third-party script inside a completely isolated, sandboxed environment. JS.JS compiles the SpiderMonkey JavaScript interpreter to LLVM bytecode with the help of Clang compiler and then use Emscripten to translate the LLVM bytecode to JavaScript. SpiderMonkey is the JavaScript engine used in Firefox web browser. Emscripten is an LLVM-to-JavaScript compiler. Fig. 2) depicts the complete process of the JS.JS technique.

JS.JS is available in the form of gzipped JavaScript library with a size of 594KB. The original library has a size of around 14MB. The authors of [14] have also provided a wrapper API which consists of 1000 lines of code and noted in their work that JS.JS has performance issues due to its heavy-weight size [14] and its nature. Mobile applications are simple in nature and have less amount of code as compared to their desktop variants. The first challenge we faced during the implementation of our example mobile mashup application was performance. The execution time, i.e. setting up the environment of JS.JS on mobile devices, is much higher (around six times) as compared to the execution time on desktop machines (220ms). Considering the factors on which we have based our evaluation, we found JS.JS does not have cross-mobile browser support. At the time of writing, JS.JS is not compatible with Internet Explorer (IE) mobile browser as it IE does not support "typed arrays". The amount of effort developers have to put in is high as the JS.JS library does not provide string manipulation operations. Developers have to rewrite the code if they want to use these functions. The developers' learning curve is also high in JS.JS case.

We believe, it is reasonable to expect that if learning curve is high then there is a great chance that developers will not use the technology due to time constraints and business pressure. Our evaluation of JS.JS on mobile devices shows that JS.JS is not reasonably suited for mobile mashup applications development.

2.2 Evaluation of OMash

Crites et al. proposed in [2] OMash for secure web-based mashup application development. The authors have modified Mozilla Firefox 2.0 for proof-of-concept implementation of OMash proposal. OMash is an abstraction that treats web pages as objects and allows objects to communicate only via their declared public interfaces. Since OMash does not rely on the same-origin policy [11] for controlling DOM access or cross-domain data exchange, it does not suffer from the

same-origin policy vulnerabilities. OMash leverages the idea of "object abstraction" from Java object model. In Java, object represents a principal and objects can communicate with other objects only via public interfaces. OMash considers every web page as principal. Contents belonging to the web page are private by default and other objects can access the content only via public methods.

OMash was available as an extenion of the Firefox 2.0 web browser. In order to demonstrate OMash, authors have to make some internal Firefox configuration settings in order to access the function "getPublicInterface" and to allow cross-domain communication. In the domain of *capability.policy.default* the three configurations to be set are *Window.getPublicInterface.get*(allAccess), *Window.getPublicInterface.set*(sameOrigin), *XMLHttpRequest.open*(allAccess). The current versions of the Firefox browser do not support these configuration values due to security concerns [9]. One alternative, we found is to reimplement the solution completely in JavaScript, but it is time consuming and will take a lot of time from developers. In JavaScript's implementation of OMash, the size of code will also grow as developers have to implement browser's security preferences explicitly in JavaScript which affects the performance of mobile mashup applications. Further, the learning curve is high and on mobile-side we do not see potential of OMash technique for mashup application development.

2.3 Evaluation of SMash

Yoshihama et al. have proposed SMash in [5]. It is a secure component model, supportingh the encapsulation of contents and data from different domains which may or may not be trustworthy. In SMash, components are linked together via communication abstraction and the ⟨iframe⟩ tag is used as a container. All mobile browsers also support the ⟨iframe⟩ tag. E.g. if a main mashup application wishes to integrate contents from three different domains, the application will use three ⟨iframe⟩ tags i.e., one for each domain. SMash is implemented in the form of an open-source JavaScript library named OpenAjax [10], which is browser independent. The model consists of components, with input/output ports, and an event hub, with mediated communication channels. A component contains contents from one domain. The event hub is a publish/subscribe system with many-to-many channels on which messages are published and distributed. Due to being open-source it is suitable for modifications and improvements.

3 SandMash: SMash with Iframe Sandbox Attribute

SMash's OpenAjax library does not support HTML5's ⟨iframe⟩ tag's "sandbox" attribute. We have added this feature to the OpenAjax library. The "sandbox" is a new attribute added to ⟨iframe⟩ tag in HTML5. It is based on a "principle of least privilege" [15]. It is now supported by major desktop browsers and according to [1], mobile browsers also support "sandbox" attribute. With the help of the ⟨iframe⟩ "sandbox" attribute, mobile mashups developers can restrict the framed-content (which may not be trustworthy) in a low-privileged environment. and they can also specify the security policy on the framed-content with the help of the following flags allow-same-origin, allow-scripts, allow-popups, allow-forms.

Listing 1.1. OpenAjax Library with "Sandbox" Support

```
var_d4=document.createElement("span");
_b7.IframeSandbox.parent.appendChild(_d4);
var_d5="<iframe␣sandbox=
'allow-scripts␣allow-same-origin␣'
id=\""+_be+"\"␣name=\""+_be+"\"
src=\"javascript:'<html>␣</html>'\"";
var_d6="";
var_d7=_b7.IframeSandbox.iframeAttrs;
```

Fig. 4. Average Performance Timing for SMash and SandMash

Testing Type	SMash	Sand-Mash
connect	391 ms	41 ms
domContent-LoadEvent	2 ms	1 ms
domainLookup	42 ms	23 ms
loadEvent	260 ms	305 ms
response-ToRequest	91 ms	47 ms

Fig. 3 shows our modified model of SMash. It includes support of the "sandbox" attribute as an additional layer of security on framed-content. In the Fig. 3, component A can publish to Channel 2 and 3 and is subscribed to Channel 1.

3.1 Implementation

In this section, we discuss our implemention of proposed SMash model for mobile mashup application development. SMash's implementation is available in the form of open source, browser-independent library. For the implementation of SandMash, we have added support of the "sandbox" attribute in the library OpenAjax Hub 2.0.7 [10]. Listing 1.1 shows a code snippet from our modified OpenAjax library and how mobile mashup applications developers can set the security policy (e.g., allow-scripts, allow-same-origin) on individual components.

3.2 Evaluation

In this section, we discuss the evaluation of proposed SMash model on our running example (see Section 3.1) by keeping in mind the factors (see Section 1). The OpenAjax library is a lightweight JavaScript library. It is reasonably suited for mobile mashup application developement. We have already discussed that our modifications in the library are also not substancial. OpenAjax is also mobile-browser-independent library. The amount of effort required by the developers is low due to the use of ⟨iframe⟩ tag for content separation. Mostly developers are aware of ⟨iframe⟩ tag so the learning curve will also be low.

It may help in adoption of modified (added layer of security in the form of "sandbox" attribute) SMash technique on mobiles. As a part of performance evaluation, we have loaded the toy mobile mashup application with original SMash library and with our modified SMash library in mobile browser fifteen times as shown in the Figure 4 above. The average load time we experienced was 260 ms for original SMash library and 305 ms for our modified SMash library.

4 Conclusion

In this paper, we evaluated three client-side Mashups proposals (i.e., JS.JS [14], OMash [2] and SMash [5]) on mobile devices. Our evaluation on mobile devices shows that the SMash proposal by IBM is reasonably suited for mobile mashups development because it requires less amount of effort from developers and at the same time it has cross-mobile-browser compatibility. We have modified the OpenAjax JavaScript library [10] proposed in SMash and have added

support of HTML5 ⟨iframe⟩ tag's "sandbox" attribute to it. With the help of ⟨iframe⟩ "sandbox" attribute, mobile mashups developers can restrict the framed-content (which may not be trustworthy) in a low-privileged environment. We demonstrated our proposal on a mobile mashup application that integrates content from three different providers (i.e., News, Stock and Weather service).

References

1. CanIUse?: Sandbox Attribute for iframes, http://caniuse.com#search=sandbox
2. Crites, S., Hsu, F., Chen, H.: OMash: Enabling Secure Web Mashups via Object Abstractions. In: Proc. of the ACM Conf. on Computer and Communications Security, (CCS'08). pp. 99–108 (October 2008)
3. Dongy, X., Tranz, M., Liangy, Z., Jiangz., X.: Adsentry: Comprehensive and flexible confinement of javascript-based advertisements. In: Annual Computer Security Applications Conf., (ACSAC'11). pp. 297–306 (2011)
4. JavaScriptinJavaScript(js.js): Sandboxing Third-Party Scripts (April 2012), http://sns.cs.princeton.edu/2012/04/javascript-in-javascript-js-js-sandboxing-third-5Cparty-scripts/
5. Keukelaere, F.D., Bhola, S., Steiner, M., Chari, S., Yoshihama, S.: SMash: Secure Component Model for Cross-Domain Mashups on Unmodified Browsers. In: Proc. of the Int. Conf. on World Wide Web (WWW'08) 2008. pp. 535–544 (April 2008)
6. Kovacevic, A., Kaune, S., Heckel, H., Mink, A., Graffi, K., Heckmann, O., Steinmetz, R.: PeerfactSim.KOM - A Simulator for Large-Scale Peer-to-Peer Networks. Tech. Rep. Tr-2006-06, TU Darmstadt (2006)
7. Liebau, N., Pussep, K., Graffi, K., Kaune, S., Jahn, E., Beyer, A., Steinmetz, R.: The Impact Of The P2P Paradigm on the New Media Industries. In: AMCIS '07: Proceedings of Americas Conference on Information Systems (2007)
8. Manyika, J., Chui, M., Bughin, J., Dobbs, R., Bisson, P., Marrs, A.: Disruptive Technologies: Advances that will transform life, business, and the global economy (May 2013), http://www.mckinsey.com/insights/
9. Mozilla: Configurable Security Policies, http://www-archive.mozilla.org/projects/security/components/ConfigPolicy.html
10. OpenAjaxAlliance: Openajax alliance open source project at sourceforge, http://openajaxallianc.sourceforge.net/
11. Ruderman, J.: The same origin policy (August 2001), http://www.mozilla.org/projects/security/components/same-origin.html/
12. Ryck, P.D., Decat, M., Desmet, L., Piessens, F., Joosen, W.: Security of Web Mashups: A Survey. In: Information Security Technology for Applications - Proc. of the Nordic Conf. on Secure IT Systems (NordSec'10). pp. 223–238 (2010)
13. spec.whatwg: HTML-The Living Standard, https://html.spec.whatwg.org/multipage/embedded-content.html#attr-iframe-sandbox
14. Terrace, J., Beard, S.R., Katta, N.P.K.: JavaScript in JavaScript (js.js): Sandboxing Third-Party Scripts. In: Proc. of the USENIX Conf. on Web Application Development (WebApps'12). pp. 95–100 (2012)
15. West, M.: Play safely in sandboxed iframes (January 4th 2013), http://www.html5rocks.com/en/tutorials/security/sandboxed-iframes/
16. Zarandioon, S., Yao, D.D., Ganapathy, V.: OMOS: A Framework for Secure Communication in Mashup Applications. In: Annual Computer Security Applications Conf., (ACSAC'08). pp. 355–364 (2008)