Security for Web2.0 application scenarios: Exposures, Issues and Challenges

Sumeer Bhola, Suresh Chari, Michael Steiner
{sbhola, schari, msteiner}@us.ibm.com
IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne

The standardization of the Document Object Model (DOM), the interfaces to access the DOM in JavaScript, and the primitives for asynchronous communication with servers has resulted in an explosion of new application models on the Internet. From a technologies standpoint, we have seen the organic growth of numerous client side programming frameworks, new data formats like JSON, and new RPC paradigms like JSON-RPC. Traditional security models which were defined and developed before these applications, technologies, and business models evolved, are simplistic, and in many cases inadequate to address the current security exposures. In this paper we document a small number of cases where we feel new security models and techniques need to be developed. To illustrate these issues we consider a typical application: a mashup consisting of content from different trust domains rendered on a single end-user browser window.

I. Interaction of content from different trust domains

The traditional browser security model dictates that content from the different domains (in different iframes) cannot interact with each other. This model, however, does not support a number of application scenarios where a controlled interaction is desirable. In fact, a number of advertising based business models depend on permitting such interactions. To overcome this restriction, interaction is sometimes enabled by proxying the different trust domains behind a single server, thereby appearing to the browser as a single trust domain. Unfortunately, now the browser security model allows the content to arbitrarily interact with each other, including reading, writing and modifying the other domain's content's DOM. This "all-or-nothing browser security model" is clearly an undesirable situation.

What is desirable is a new security model that directly allows content to be separated by trust domains with a carefully mediated interaction between content from different domains. A promising approach that can be used to bootstrap enforcement of the isolation issues is to introduce new tags that serve to delineate content from different trust domains [1]. However, from an application programmer's perspective, we think that there must be a higher level programming interface that makes creating mashups easier and more secure. A desirable feature in this interface by hiding interaction patterns of reading, writing and modifying DOM level constructs with mediated interaction at the level of provided services. Thus, sophisticated security policies can be enforced using underlying browser defined enforcement primitives. Permitting this cross-domain interaction would help programmer avoid a number of extremely unsafe programming mechanisms such as JSONP [2], where a script downloaded from a remote domain is executed without mediation. In the remainder of this section we sketch a proposal for a secure component model using an event hub for inter-component communication, and describe some implementation challenges.

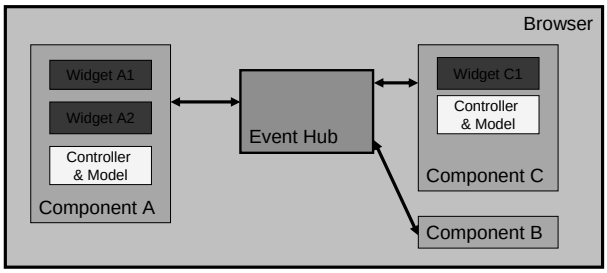

Fig. 1. Secure component interaction using an event hub

A. Secure Component Model

Content from different trust domains is encapsulated as components with typed input and output ports. The ports of a component represent an asynchronous event/message passing interface offered by that component. The mashup developer chooses the components to use in the mashup, and wires them together. The wiring represents typed communication channels such that each channel connects one output port with one or more input ports. The channels are implemented using an event hub, that is part of the trusted computing base (figure 1). Channels may be named, so the event hub is effectively implementing a publish/subscribe model of communication with topics naming the channels. Dynamic creation of new channels at runtime could be supported by a component creating a new input channel, and giving publish access to other components.

B. Implementation Challenges

Realizing this model in the context of current browsers is challenging since we have to ensure that the component

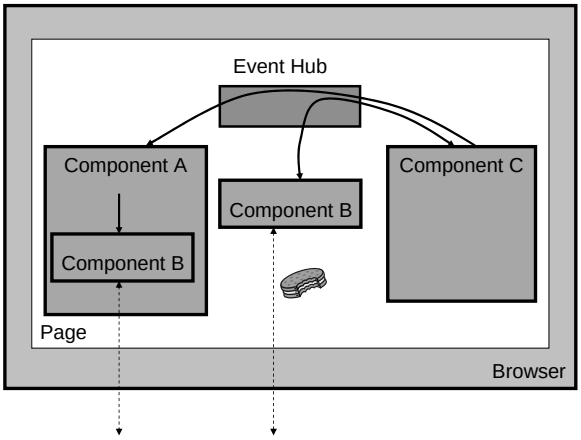

Fig. 2. Confused deputy problem with cookies and nested components

interface is the only means of inter-component interaction in the browser. One approach is to use static analysis and code rewriting [3]. Another approach we are investigating is using iframes for isolation. In this technique each component is placed in an iframe from a different domain, and inter-iframe communication occurs using fragment identifiers that are part of the iframe URL. This approach for inter-iframe communication has been previously used in Dojo cross-domain XMLHttpRequest [4]. Note that we can use the same-origin security policy of current browsers to securely compartmentalize content from different trust domains coming from the same origin server by serving them from different ports or using different addresses for the same server.

Inter-component communication can also occur through persistent state in the browser, specifically cookies. With appropriate selection of component domains, cookies will not be shared between components from different trust domains, however this does not eliminate occurences of the confused deputy problem. Figure 2 shows an example, where the application page contains the trusted event hub, and the page loads components A, B, and C in separate iframes. Component B offers an interface to make remote calls, which require a user credential represented as a cookie. The event hub only allows component C to use this remote call interface of component B. However, a malicious component A can load a copy of component B in a child iframe, which has access to the same cookie, and then use this copy to make remote calls. Double submission of cookies, which is currently used to avoid the confused deputy problem in the web (cross-site request forging), does not fix the example we have described.

II. End-user experience of security and policy

From an end-user perspective, current security models are difficult to understand, impossible to configure and the resulting click-through behavior has resulted in a major outbreak of problems such as phishing. We feel that this will be further exacerbated in new application models where content from different trust domains are rendered at the user's browser at the same time. Currently deployed methodologies and protocols are largely inadequate to address any of these challenges.

End-user experience is particularly important in the context of authentication: When prompted for authentication credentials in a composite mashup application, how is the context for this authentication conveyed to the user? Can we make authentication protocols that can safely carry password information deployable in these contexts? This problem of securely delineating the various trust domains at the user interface is different from the traditional definition where the user does not trust the device used to enter credentials.

III. New technologies resulting in new exposures

One characteristic of the new class of applications is the organic growth of new data formats like JavaScript Object Notation (JSON) [5], and the popularity of "lightweight" web services based on principles of REST [6]. This is in contrast to more standardized web services based on XML and SOAP. Security models and protocols for SOAP-XML-RPC have been extensively studied [7] and a number of protocol primitives, profiles and techniques are available which can be used to secure many interaction scenarios including multi-hop calls. While RPC based on the principles of REST and data formats like JSON are appealing due to their inherent simplicity, there has been very little effort to incorporate or define any form of security at a messaging level. There are isolated examples where some effort has been made to address security at a message level [8] but in most cases the only security used is TLS, at the transport level. It is imperative that proven security mechanisms and protocols are used with these new RPC paradigms and data formats.

Thus, current security models and primitives are inadequate to address the demands of the new application models, protocols and technologies. We believe that new security models are needed to cover the exposures in this paradigm.

IV. Acknowledgements

We would like to thank Larry Koved for many useful discussions on the topics of this paper.

References

[1] D. Crockford, "The <module> tag proposal," Described online at http://json.org/module.html.
[2] "Jsonp," An informal description is at http://ajaxian.com/archives/jsonp-json-with-padding.
[3] K. Vikram and M. Steiner, "Mashup component isolation via server-side instrumentation," in *Proceedings of W2SP 2007 (Web 2.0 Security and Privacy Workshop)*, 2007.
[4] "The dojo toolkit," Available online at http://dojotoolkit.org/.
[5] D. Crockford, "The javascript object notation," Described online at http://www.json.org.
[6] R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.
[7] "The oasis web services security(wss) tc," Specifications available online at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.
[8] "Amazon web services request authentication," Described online at http://docs.amazonwebservices.com/AlexaWebInfoService/1-0/RequestAuthentication.html.