

# The Browser as a Secure Platform for Loosely Coupled, Private-Data Mashups

Ben Adida\*

A *web mashup* is a combination of data from disparate web-based sources. Typically, a mashup loads data from pre-determined sources, “makes sense” of the different datasets in order to relate them, and serves up a useful combination. Because of browser security constraints, these mashups are usually limited to public data sources, e.g. Google Maps, Craigslist or public RSS feeds.

**Vision: Loosely Coupled Private-Data Mashups in the Browser.** We suggest that mashups will eventually combine public *and multiple private* data sources, e.g. Google Maps with a user’s calendar, address book, evites, etc. These mashups will execute inside the user’s browser, with the mashup code downloading private data using the browser’s JavaScript API. Some of these mashups will remain tightly coupled: the mashup code will designate precisely which sources can be combined. However, as interoperable metadata markup gains popularity, loosely coupled mashups will prove to be vastly more powerful: a user will combine data from various sources opportunistically, at will and without prior planning. The beginnings of this vision—and a small hint of its power—can be seen in `del.icio.us`, Magnolia, StyleFeeder, and other such applications where a user can annotate and see friends’ annotations on *any* web page.

We suggest that, rather than hard-wiring into the browser specific functionality, such as identity management, address-book-or-calendar integration, bookmark synchronization, etc., the browser should focus on becoming *a more secure and more powerful platform for mashups*, at which point specific functionalities will be built as natural and modular applications within this platform.

**Complications.** A world of loosely coupled web mashups presents significant security challenges. How does one mash up two mutually distrusting sources? How can the user enable specific data combinations but not others? How can the user provide a subset of her private data to a mashup without exposing extraneous private data? If these questions are not satisfactorily answered, developers will implement insecure “hacks”: many social networking sites today support Google, Yahoo, or Hotmail address-book integration by prompting the user for her password and impersonating her to the address-book-storing web site.

In current browsers, two techniques are used to achieve very basic loose coupling: bookmarklets and browser plugins. A bookmarklet is a snippet of JavaScript that, when clicked, is injected into the current page scope. A plugin is an extension to the browser which can choose when and how to “intervene.” Given the extremely dynamic nature of JavaScript, bookmarklets are limited in what they can safely achieve: the APIs they depend upon may be hijacked at any point. At the other extreme, a user must inherently trust all installed plugins with complete control of her

---

\*Center for Research on Computation and Society, Harvard. Email: `ben@eecs.harvard.edu`.

browser. This is highly unsatisfactory: we suggest three browser improvements to obtain a *more secure platform for loosely coupled, private-data mashups*.

**Improvement #1: Isolated JavaScript Environments.** The unstable API security problem was encountered by the GreaseMonkey plugin in 2005: a malicious web page could hijack the GreaseMonkey calls and gain access to its higher-privilege API. The problem was fixed by giving GreaseMonkey scripts a “clean-slate” JavaScript environment, including its own direct access to the DOM. We propose the same tool as a generic JavaScript construct, e.g.:

```
with_cleanslate { ... }
```

This proposal will allow different JavaScript components to be combined dynamically without conflict, effectively providing isolation without additional privileges.

**Improvement #2: Extensions with Fine-Grained Permissions.** We propose that browsers implement an API for *limited extensions*, whose power lies somewhere between a bookmarklet and a plugin, both in accessing browser data and making additional network requests.

- **Limited Awakening:** a limited extension would only be “awakened” on certain pre-determined hostnames, or when the user explicitly activates it, e.g. using a button or menu item. If a user invokes a limited extension on a new hostname, she would then have the option to automatically awaken the extension on future visits to that same host.
- **Limited Network Requests:** a focused extension should be able contact the hostname on which it is currently awakened *and* the extension’s “home domain,” but no other source. Extensions may be granted the right to contact additional hosts at the user’s discretion in the same way that the extension is granted awakening privileges.

Thus, a user can designate site-specific browser extensions, trusting these extensions only for certain well-defined facets of her life, and enabling private-data mashups between certain sites only.

**Improvement #3: Metadata-Mediated Extensions.** We propose that browsers implement an interoperable structured data mechanism for private exchange, or, in simpler terms, *autofill on steroids*. Consider, for example, the New York Times’ “recommend this story to a friend” feature. The NYT should be able to request, in a structured way, “email addresses from the user’s address book,” at which point the browser can trigger the proper metadata-mediated extension that handles the user’s contacts, either from a local file or from web-based storage. The extension leads the user through the selection of these contacts, and eventually returns the private data to the NYT scope, which can then proceed with its feature. The user achieves true private data integration, while the NYT receives no more data than it needs to achieve the user-requested goal.

**Conclusion.** Innovation on the web has always come from loose coupling, and mashups likely will be no different. Vast power lies in enabling opportunistic combinations of features and private data. Today, developers are achieving such functionality via patchy, unstable solutions or omnipotent extensions with minimal security considerations. In some egregious cases, users are being implicitly trained to give away their passwords when the slightest data integration is required. To correct this trend before it gets worse, the browser should become a platform for opportunistic mashups with solid security policies: isolation, fine-grained permissions, and interoperable metadata mediation.