Mashups (continued)

Challenge comes when browser has credentials and the web services need those credentials to run its services.

Option 1. (undesirable)  Tell the mashup your credentials to both services

Option 2. (desirable) Have your browser use the credentials separately to each service.

* But Here we run into same origin policy, where JS from one server can’t use credentials for or from another service.

You can put a `<script src = serviceA.com/......>` in your mashup page. That will load the JS, but the same origin policy still applies and the script is still run in the domain of the mashup.

This doesn’t fully solve the problem^^

Modern Solution: Iframe

**Iframe**
- Like a separate web page within a webpage
- Like a Dynamically Linked Library
- No interaction through browser

Facebook Connect Example: Facebook announcement = “In August 2006, we introduced the first version of the Facebook API, enabling users to share their information with the third party websites and applications they choose. Hundreds of companies have leveraged these APIs, allowing users to dynamically connect their identity information from Facebook, such as basic profile, friends, photos information and more, to third party websites, as well as desktop and mobile applications.

In May 2007, we launched Facebook Platform, which allowed third party developers to build rich social applications within Facebook. More than 350,000 developers and entrepreneurs from 225 countries have signed up, and started developing applications, and have seen significant adoption by Facebook users worldwide.

Facebook Connect is the next iteration of Facebook Platform that allows users to "connect" their Facebook identity, friends and privacy to any site. This will now enable third party websites to implement and offer even more features of Facebook Platform off of Facebook – similar to features available to third party applications today on Facebook.”
**Post Message (HTML 5)**
- Allows Iframes to send each other messages

So you can have a page with an outer iframe and an inner iframe, (Mashup.com is outer, facebook.com is inner)

The inner iframe can receive a message from the outer one and reply with a result. This is a way to allow Mashup.com to send requests to facebook.com (defeating same origin policy) in a controlled way.

Combines privilege and policy. ^^

Facebook.com iframe needs to authenticate the outer mashup.com iframe

The Browser has to tell the receiving iframe where the message is coming from.
- browser provides ID of message sender. (Similar to IPC setup like Binder)

Suppose you have another inner iframe for bing.com, These 2 inner iframes want to pass messages but can’t trust the outer iframe. Then they can create a socket between the bing.com iframe and the facebook.com iframe to directly communicate.

*So the conclusion is don’t try to do mashups w/o HTML 5 Post Messages*

-----------------------------------------------

**Defenses**

Happens at many stages of the software pipeline.

- **Design level**
  - EX: Separation of privilege

- **Coding level**
  - EX: ban SQL.query
  - EX: Code auditing

- **Testing**
  - EX: Fuzzing -
  - EX: Red Teaming

- **Prevent Bug Exploit**
  - EX: Once software Ships, make use of things like ASLR and ASCII Armoring

- **Damage Mitigation - may not prevent exploit but will limit damage they can do**
  - EX: Intrusion Detection System
  - EX: Chroot Jails
  - EX: VM’s