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

*Sumeer Bhola, Suresh Chari, Michael Steiner*
Storyboard for motivation slides

- **Setup:**
  - Security-conscious developer on drawing board for used-car mashup
    - Services offered:
      - goohooque (map)
      - Dealers (generally: address, inventory via REST/JSON, dealer-specific details for example as active (link to KBB) HTML)

- **Sequence:**
  - Goes through various cycles & rejects each for functional or security problems, finally gives up …
    1. Get info for list via Direct XHF -> non-functional: same origin
    2. Try with ScriptSrc -> no security: complete access to DOM & user credentials
    3. Try with Proxy -> controlled (data) service access …
       … but what about rich-text dealer info? -> ahh, ACF?! …
       … yet dealer wants to mash up with to KBB & allow price negotiation -> aargh, no security on active component
    4. Ok, get via iframe -> secure but no way to synchronize update table for negotiated price?!
    5. Give up frustrated …

- **Observations:**
  - Very hard to find security solutions; very context/deployment specific!!
  - Most developers would not even have realized problems; insofar, above idealistic scenario!!

- **Message:**
  - We need to give developer a ``tool” which is
    - fail-safe (secure-by-default),
    - easy-to-use (otherwise not used) &
    - deployment setup-independent (important for mashup component providers)
Once upon a time there was a mashup developer …
Same Origin does not allow
Give up complete control?!

<table>
<thead>
<tr>
<th>Price</th>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Options</th>
<th>Dealer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$</td>
<td>Chevy</td>
<td>01</td>
<td>Blue</td>
<td>Sunroof</td>
<td>MoCo</td>
</tr>
<tr>
<td>$</td>
<td>Lada</td>
<td>77</td>
<td>Red</td>
<td>Urkidn</td>
<td>Cheapo</td>
</tr>
<tr>
<td>$$$</td>
<td>Rolls</td>
<td>93</td>
<td>Gold</td>
<td>All</td>
<td>ShDy</td>
</tr>
</tbody>
</table>

Map Srv

Mashup Srv

Map Srv

Dealer Srv

Detail Car info

<script>
Rich Text with Active Content?!
How update negotiated price in list?
... and he lived for a long time afterwards, unhappy and problem unsolved.
A Fairy Tale?

Yes …

… but not because problem not hard --- we have not even talked about authentication & credentials --- and very deployment-sensitive ..

… but because most developers would not even have realized all problems!

Therefore ..

… we need to give developer a “tool” which is

– fail-safe (secure-by-default),

– easy-to-use (otherwise not used) &

– deployment setup-independent (important for mashup component providers)
``Tool” requirements

- Identity propagation
- Data integrity
- Usability
- Security
- Privacy
- Policy
- Domains

• **Foundation:** Isolation/separation & mediation

• Make it easy to write secure component interactions
  • Secure by default

• Authorization Policies must be consumable
  • Base on simple abstractions: service interface as opposed to DOM level
  • Declarative to allow for separation of concerns
  • Closer in spirit to J2EE than to J2SE …

• Minimize assumption on end-user
Secure Component* Model: Approach

Related concepts:
- OpenAJAX Hub
- Module Tag proposal
- Dojo publish/subscribe
- UI (widget/view) and/or logic (model/controller)
- Interacts with other components only using Event Hub
- No shared resources!
- Trust Domain associated with component
- Usability of security: high-level security policy specification
- UI separation for components
- Lifecycle management of components
- Policy for dynamic component creation
- Message passing abstraction
  - High-level abstraction that can be implemented using multiple low-level mechanisms
  - Publish/subscribe channels: topics for channel naming and support for many-to-many channels
- Enforces security policy on all inter-component interaction
- Security Policy expressed per message channel (send/receive access on that channel)

* Pick your favorite alternative name: Gadget, Widget, ..
Secure Component Model: Prototype

- Enforcement of component boundaries: Using `<iframe>` isolation and fragment ids for parent-child frame communication
  - Event Hub implemented by main application frame
    - provided by mashup maker
  - Mashup maker is trusted to define inter-component communication

- Channel Policy
  - Mashup maker defines static inter-component message channels when loading components
  - Dynamic channels only permitted between components with compatible labels

- End-to-end security
  - Component credential in addition to user credential
  - Unified and CSRF-resistant request authentication
End-user Experience of Security

Content from multiple domains on a page increases existing problems

- Theft or Misuse of user credentials (Phishing++, CSRF++)
  - Theft: Browser URL address bar is useless for mashups
    - Does not communicate context of authentication challenge to user – which credential should be given?
    - Need integrity of context and fail-safe protocols
      - e.g. Identity Selector for Windows CardSpace or Higgins Trust Framework, pwd-based key-exchange protocols

- Confidentiality of input and integrity of display
  - Need to securely delineate different trust domains or components in the user interface
  - How does the user know where its input is going
  - Where parts of the display are originating?

- Who are the stakeholders in defining security policy?
  - Mashup maker (a.k.a. man-in-the-middle) not necessarily trusted with user credentials
    - With browser support, need not trust mashup maker with confidentiality of input and integrity of display
  - Mashup maker deciding inter-component wiring policy
    - With browser support, can the user / component providers get more control of this policy?
      - How are these policies defined? Enforced?
Summary

- Current security models are inadequate for Web 2.0
  - Browser models are either too restrictive or permissive
  - Built on brittle ground (DNS, cookies, ..)
  - Workarounds lead to unsafe practices

- Need new security models to address the new application paradigms
  - End-to-end isolated components
  - Explicit and mediated component interaction
  - End-user experience and credentials

- Migration path
  - Need secure (but enforcement-independent) programming model now!!
Building Blocks for Enforcement

- **Browser extensions**
  - `<module>` tag proposal [Crockford]
    - Components talk ONLY through a send/receive interface
    - Could consider extending with each module exporting a list of allowed functions
  - DOM access control
    - Each component comes with a dom level access control policy
  - FRIV element proposal (MashupOS) (Microsoft Research)

- **Server-side code instrumentation**
  - Static analysis & code rewriting
    - BrowserShield [Microsoft Research]
    - Vikram and Steiner [IBM Watson]
    - Secure DOM Javascript Library [IBM Tokyo Research]
  - Safe language with code translation
    - e.g., GWT with security guarantees,

- **Iframe isolation: server-managed DNS sub-domains (virtual server) per colocated components**
  - Inter-iframe communication using e.g. fragment ids (dojo), document.domain (crockfort, Subspace), applet

*Actual technique (or combination thereof) allows for trade-offs and deployment time adaptation*

*Chosen (declaratively) according to setup & trade-off between security & performance*