OMash: Enabling Secure Web Mashups via Object Abstractions

Steven Crites, Francis Hsu, Hao Chen
UC Davis

Mashups and the Same Origin Policy

- Mashups integrate content from multiple websites
- Content protection relies on Same Origin Policy (SOP)
  - Currently, contents get complete or no isolation
  - MashupOS proposes more flexible trust relationship [SOSP 07]
    - Isolated
    - Open
    - Access-Controlled
    - Unauthorized

Same Origin Policy

Server

Browser

a.com

b.com

Problems with SOP – What Domains are of the Same Origin?

- DNS Insecurity
  - Client vulnerabilities
    - DNS rebinding (Jackson et al, CCS 07)
    - Dynamic Pharming (Karlof et al, CCS 07)
  - Server vulnerabilities
    - DNS cache poisoning (Kaminsky, BlackHat 08)

Cross-Site Request Forgery

Server

Browser

a.com

b.com
OMash: Object Mashup

• A new browser security model
• Use Object-Oriented model (e.g. Java object model)
• Treat each Web page as an object
  – Encapsulate all scripts and data
  – Objects declare public interface
  – Objects communicate only via public interface

Object Abstractions

• Java (analogy)
  public class FooObject {
    public void publicMethod() {
    }
    private int privateData;
  }

• Web page object
  <html>
  <script>
    function getPublicInterface() {
      function Interface() {
        this.publicMethod = function () {
          ...
        }
        return new Interface();
      }
      return new Interface();
    }
    var privateData;
  </script>
  </html>

Page Objects

• A page consists of
  – DOM tree
  – Scripts
  – Credentials (HTTP auth, cookies)
• A page object can be contained in a
  – Window
  – Tab
  – Frame
  – Iframe

Public and Private Members

• Public interface
  – Each object declares getPublicInterface()
  – Returns a closure of all public methods and data
• Private data
  – DOM
  – Scripts
  – Credentials

Usage Example

• map.html
  <html>
  function getPublicInterface() {
    function Interface() {
      this.setCenter = function (lat, long) {
        ...
      }
      return new Interface();
    }
  }
  </html>

• integrator.html
  <iframe src="map.html">
    ...
    var map = win.getPublicInterface();
    map.setCenter(lat, long);
  </iframe>

Trust Relationships

• Can model trust relationships needed for mashups (as identified by MashupOS)
  – Isolated
  – Open
  – Access-Controlled
  – Unauthorized
Isolated

• No access between provider and integrator

```javascript
function getPublicInterface()
{
    function Interface()
    {
    }
    return new Interface();
}
```

Open

• Full access between provider and integrator

```javascript
function getPublicInterface()
{
    function Interface()
    {
        this.getDocument = function ()
        {
            return document;
        }
    }
    return new Interface();
}
```

Access-controlled

• Limited access depending on caller

```javascript
function getPublicInterface() {
    function Interface() {
        this.auth = function(user, pass) {
            return token;
        }
        this.do = function (token,...) {
            check(token);
        }
    }
    return new Interface();
}
```

Preventing CSRF

Server

```javascript
var api = win.getPublicInterface();
```

Browser

```javascript
api.auth(user, pass);
```

Preventing CSRF

Server

```javascript
api.do(token,...)
```

Browser

```javascript
No cookie!
```
Browser Sessions under OMash

• Each cookie
  – belongs to a window
  – is shared by subsequent pages from the same domain in that window
• Each window has an independent session
  – Desirable side effect:
    Can log in to multiple accounts in different windows in the same browser

Cross-window Sessions

• How to track a session across windows?
• Cookie Inheritance
  – When page P1 loads P2, P2 inherits P1’s cookies
  – P1 and P2 now belong to the same session

Implementation

• Proof of concept as Firefox add-on
  – Make an exception to SOP in Mozilla’s Configurable Security Policy
  – Change Cookie Manager to make each cookie private to a window
• No changes required on the server

Supporting SOP without DNS

• If application prefers using SOP to allow inter-page communication:
  – Server embeds a shared secret in all pages
  – Pages authenticate each other using this secret

Related Work

• MashupOS (Wang et al, SOSP 07)
• SMash (Keukelaere WWW 07)
• Google’s Caja

Supporting SOP without DNS

```javascript
<script>
  secret = "1234"
  function getPublicInterface() {
    function Interface() {
      this.foo = function (secret, ...) {
        check(secret); ...
      }
      return new Interface();
    }
    return new Interface();
  }
  api = win.getPublicInterface()
  api.foo(secret, ...)
</script>
```
Conclusion

• OMash a new browser security model
  – Allows flexible trust relation
  – Simple
  – Familiar, easy to understand
• Don’t rely on Same Origin Policy
  – Prevent CSRF attacks
  – Allows programmers to define “Same Origin” flexibly based on shared secrets