Towards Building Secure Web Mashups

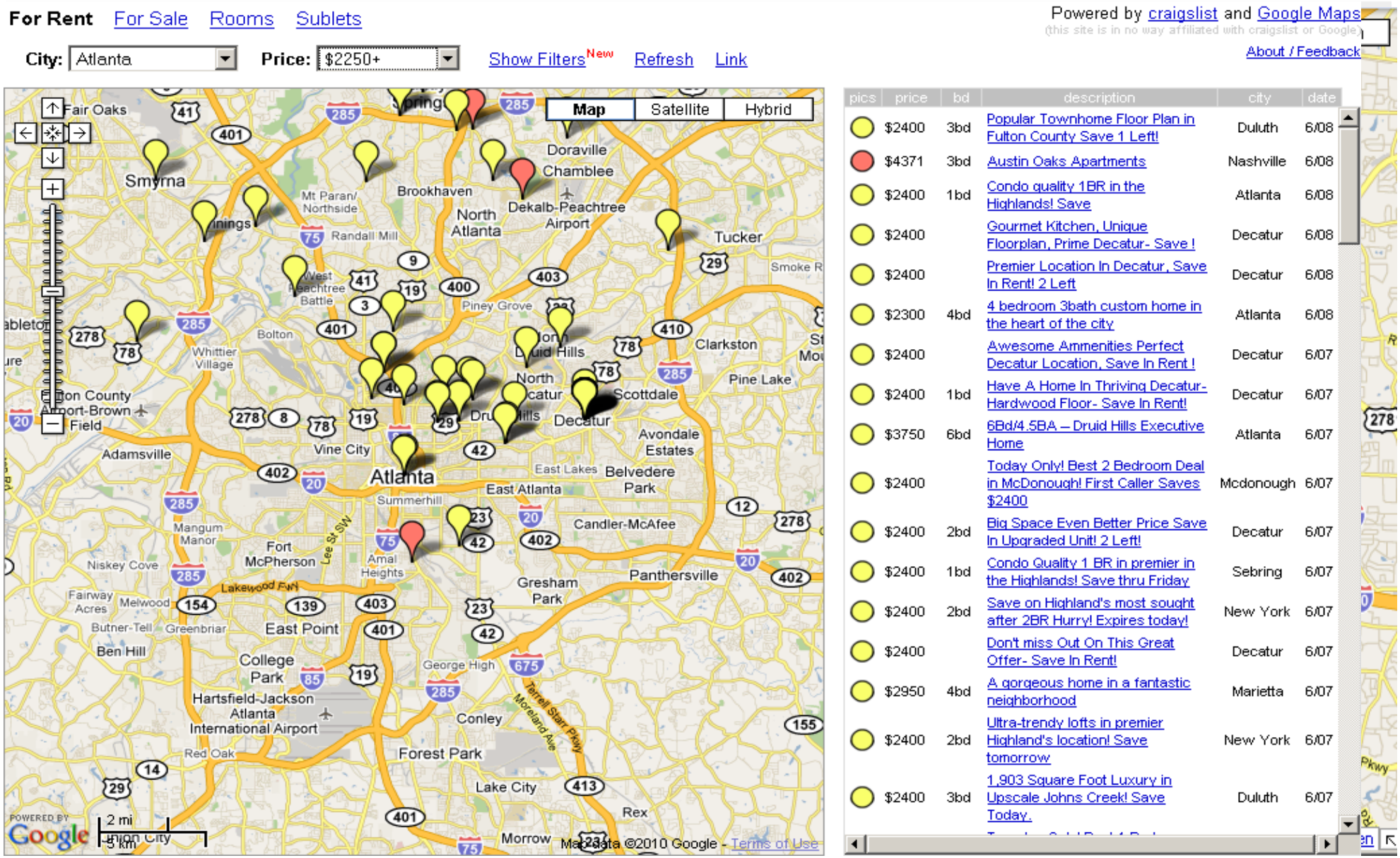
Maarten Decat
Philippe De Ryck
Lieven Desmet
Wouter Joosen
Frank Piessens

DistriNet Research Group
Katholieke Universiteit Leuven, Belgium

OWASP
23/06/2010

The OWASP Foundation
http://www.owasp.org

Mashups by example



$440 / 2br - WERE ELSE DO YOU GET A 2BDR FOR THE PRICE OF A 1 BDR?????? - (MARIETTA) img

Mashups: Definition

A web application that combines content (data/code) or services from multiple origins to create a new service

Incentives for mashups

- Added value of combined result
- Content re-use
- Flexible and lightweight applications

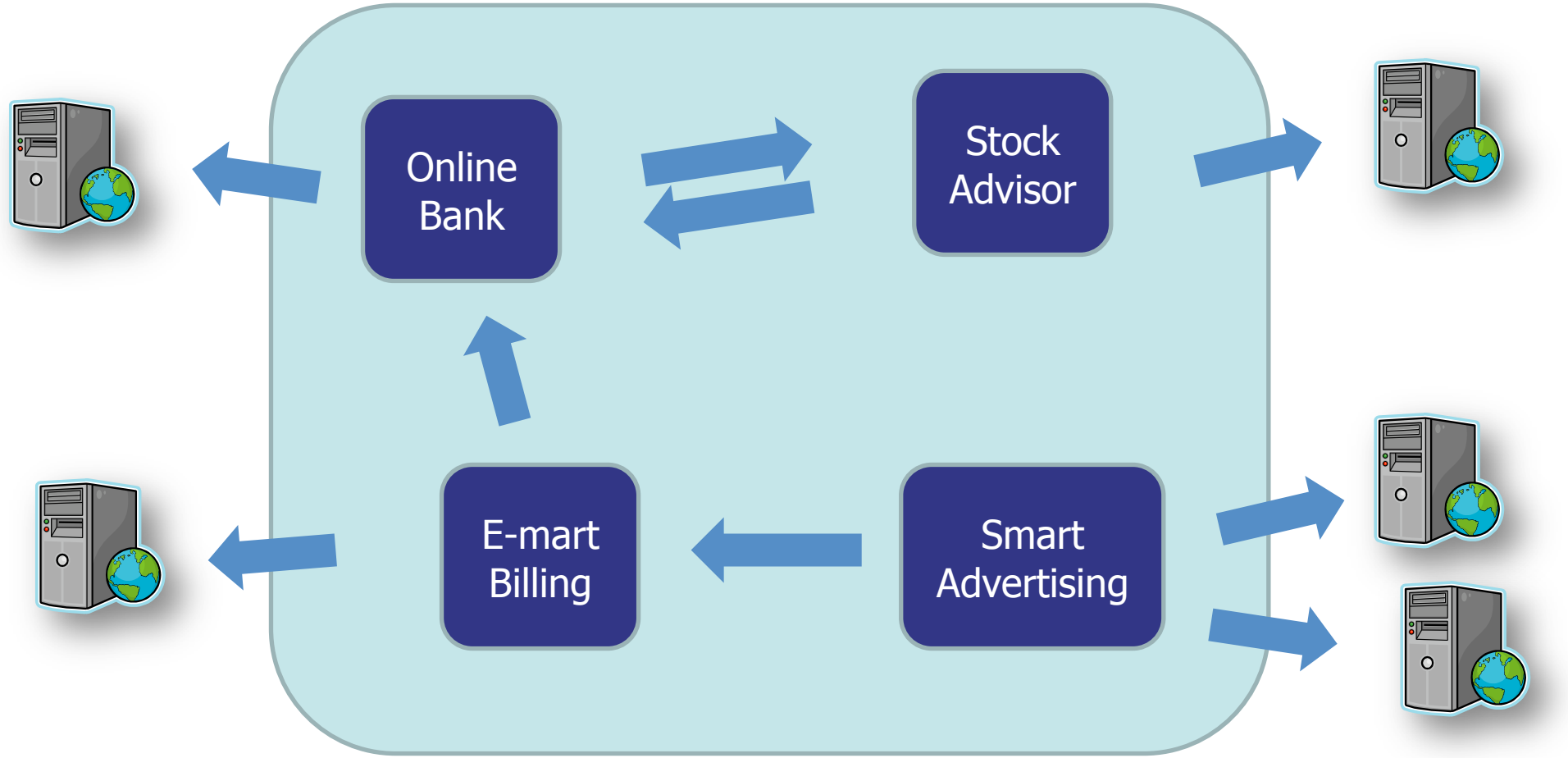
Presentation Overview

1. Mashup Requirements

2. Mashup Security
   - Separation
   - Interaction
   - Communication

3. Future Developments

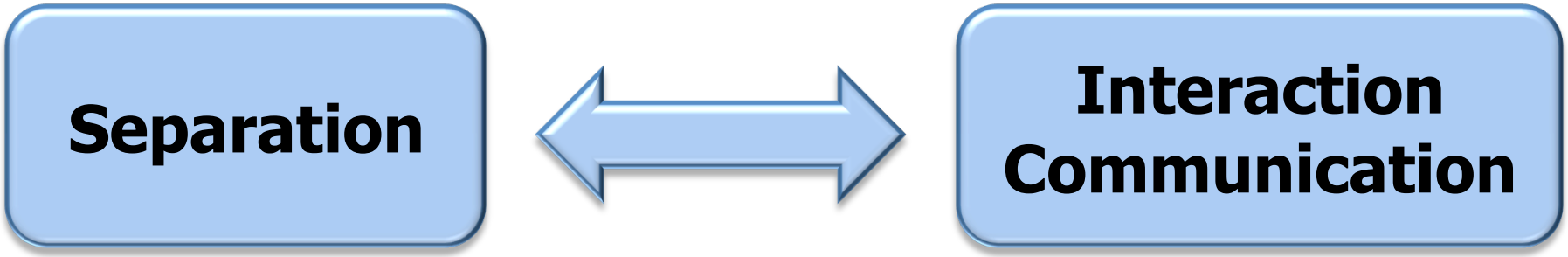
Example Case: The Financial Mashup



Requirements for mashups

- Interaction with other components
- Communication with integrator / provider

- Data / code protection
- Restricted interaction



Same Origin Policy

- Basic security policy of the web
  - Constructed for static applications
  - Separates documents from different origins
  - Limits communication to document origin
- SOP and HTML
  - IFRAME offers document separation using domains
  - SCRIPT offers script inclusion and interaction
- Insufficient for dynamic mashup applications

Leveraging separation (1)

- Restriction of the SOP
  - No interaction between different-origin documents
- Mashups have a history of enabling interaction:
  - Fragment Identifier Messaging [1]
  - SMash [2]
  - Subspace [3]
  - postMessage [1]

Leveraging separation: postMessage

- Enables frame communication
  - JavaScript API to send/receive messages
  - Event-driven
  - Mutual authentication
- Standardized
  - Part of HTML5
  - Already supported in major browsers

```
window.addEventListener("message", rcv, false);

function rcv(event) {
  if (event.origin !== "http://example.org") return;

  //handle event
}
```

```
var f = frames[1];
f.postMessage("abc123", "http://frame.example.com");
```

Leveraging separation (2)

- Restriction of the SOP
  - No separation between same-origin documents
- Stronger separation than IFRAMES:
  - Module-tag [4]
  - MashupOS [5]
  - OMash [6]
  - Sandbox-attribute [7]

Leveraging separation: sandbox

- Provides frame restrictions
  - Unique origin
  - Disable plugins, forms, script, navigation
- Standardized
  - Part of HTML5
  - Not yet supported in major browsers (only Chrome)
- Some underspecified behavior
  - Unique origin and cookies
  - Unique origin and interaction/communication

```
<iframe src="http://example.com" sandbox >…</iframe>
```

Leveraging interaction (1)

- Restriction of the SOP
  - No separation between loaded scripts (origin agnostic)
- Restriction of script inclusion
  - No control over loaded scripts

- Subsetting JavaScript:
  - ADSafe [8]
  - FaceBook JavaScript [9]
  - Caja [10]

Leveraging interaction: Caja

- **Goal**: object-capability security in JavaScript with a minimal impact
  - Static verification
  - Runtime checks
- Allows reasoning about the language [11]
- Successfully used on Yahoo Application Platform, iGoogle, …

Leveraging interaction (2)

- Restriction of the SOP
  - No separation between loaded scripts (origin agnostic)
- Restriction of script inclusion
  - No control over loaded scripts

- Behavior control / Policy enforcement:
  - Browser Enforced Embedded Policies [12]
  - Self-Protecting JavaScript [13]
  - ConScript [14]
  - Secure Multi-Execution [15]

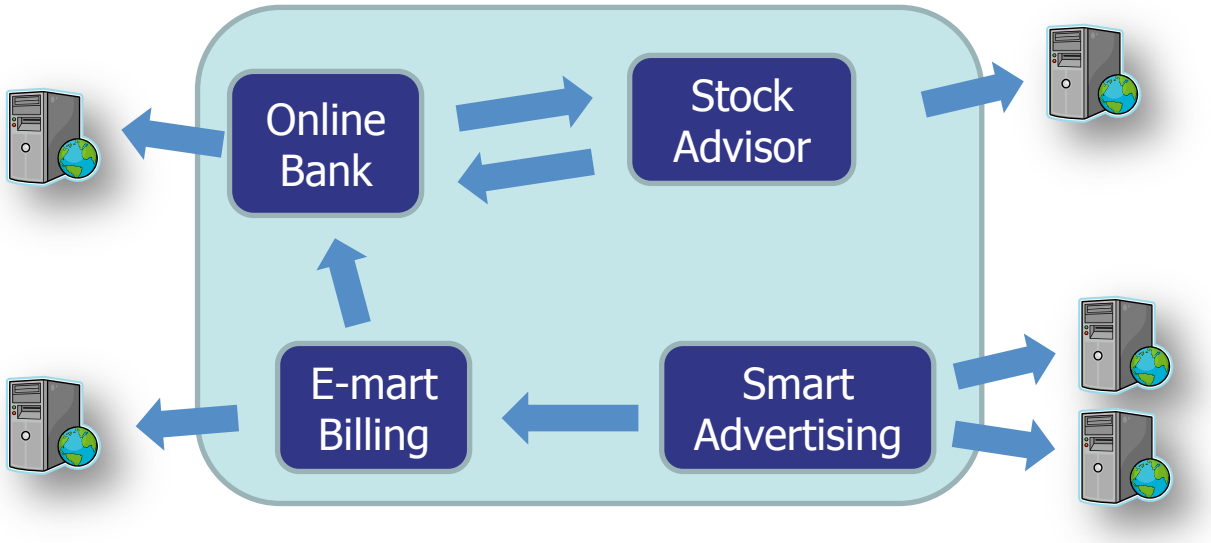
Enabling Communication

- Restriction of the SOP
  - No communication to different origins
- Mashup techniques have proven otherwise:
  - Client/Server-side Proxies [3]
  - Script Communication
  - Plugin Communication (Flash, Java, …) [16]
  - Cross-Origin Resource Sharing [17]

Enabling Communication: CORS

- Enables cross-domain communication
  - Same mechanism as XHR
  - Uses additional headers to supply information
  - Enforcement by browser
  - Protection of legacy code!
- About to be standardized
  - W3C Working draft
  - Specifies API and algorithms, not implementation
  - Already supported in major browsers

Overview



Data / code protection: sandbox / caja

Interaction with other components: postMessage

Communication with integrator / provider: CORS

Restricted scripts: caja / policy-based techniques

Future of mashup security

- Mashup situations are extremely complex
  - Current techniques are strong foundation, but need abstractions to become powerful
- (Business) requirements
- Policy based approach
  - Provided with the application
  - Controls fine-grained aspects (isolation, restriction, …)

References (1)

[1] *Securing Frame Communication in Browsers*, Barth, A. et al, 2008

[2] *SMash: Secure Component Model for Cross-Domain Mashups on Unmodified Browsers*, De Keukelaere, F. et al., 2008

[3] *Subspace: Secure Cross-Domain Communication for Web Mashups*, Jackson, C. et al. 2007

[4] *The <module> tag*, http://www.json.org/module.html, 2010

[5] *MashupOS: Operating System Abstractions for Client Mashups*, Howell, J. et al., 2007

[6] *OMash: Enabling Secure Web mashups via Object Abstractions*, Crites, S., 2008

[7] *HTML 5 Working Draft*, Hickson, I. et al., 2010

[8] *ADSafe*, http://www.adsafe.org/, 2010

[9] *FBJS - Facebook Developer Wiki*, http://wiki.developers.facebook.com/index.php/FBJS, 2010

References (2)

[10] *Caja: Safe active Content in Sanitized JavaScript*, Miller, M. et al., 2008

[11] *Object Capabilities and Isolation of Untrusted Web Applications*, Maffeis, S. et al., 2010

[12] *Defeating Script Injection Attacks with Browser-Enforced Embedded Policies*, Jim, T., 2007

[13] *Lightweight self-protecting JavaScript*, Phung, P. et al., 2009

[14] *ConScript: Specifying and Enforcing Fine-Grained Security Policies for JavaScript in the Browser*, Livshits, B. et al., 2009

[15] *Noninterference Through Secure Multi-Execution*, Devriese, D. et al., 2010

[16] *Browser Security Handbook*, Zalewski, M., 2010

[17] *Cross-Origin Resource Sharing*, van Kesteren, A., 2009