



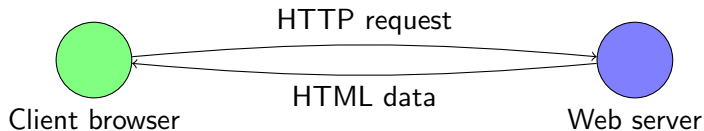
Coping with a mash-up of threats in web applications¹

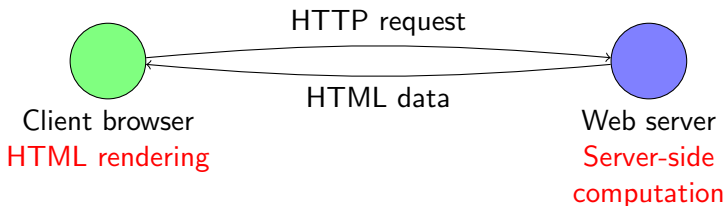
Mattia Monga

Dip. di Informatica e Comunicazione
Università degli Studi di Milano, Italia

mattia.monga@unimi.it

Saint-Jean-Cap-Ferrat — May 5, 2009





This is a typical client-server setting, in which the threats are mainly for the server's assets (the enemy is the client)

Mash-up
threats

Mattia Monga

Web
applications

The old web

The rising of the
client side

Mash-ups

Highlights of
current
research

An open
(brave) new
world

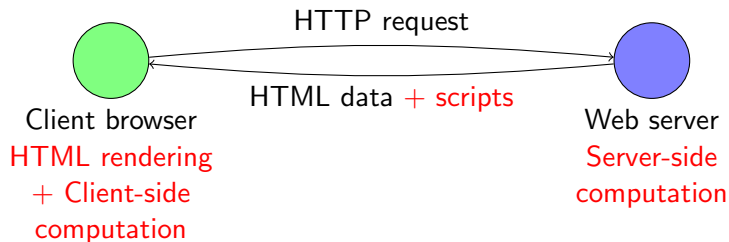
Conclusions

Protecting this kind of web mainly means **discover server's vulnerabilities** to prevent:

- SQL and file injections
- Session forgeries
- Race conditions
- ...

Some tools exist to perform **static** or **dynamic** analyses on the server application code.

The web, revisited



Current Rich Internet Applications rely on a **code on demand** architecture, in which the client executes scripts coming from the server together with HTML. Browser and user's data might be at risk.

Mash-up
threats

Mattia Monga

Web
applications

The old web

The rising of the
client side

Mash-ups

Highlights of
current
research

An open
(brave) new
world

Conclusions



Again, most of the work is done on the server side to avoid Cross-Site Scripting (XSS) injections.

The client is basically protected by two measures:

- 1 **sand-boxing** of all the scripting code: sometimes too generic to be effective (e.g., cookie stealing, history disclosure, . . .);
- 2 **same origin policy**, scripts coming from different domains cannot interfere: sometimes too strict (more on this later)!

Who is the enemy of the client? Maybe other clients (XSS), or **server**?

A trivial example



A password manager online: it advertises that only encrypted data are saved on its servers.

clipperz keep it to yourself

logout lock cards account data bookmarklet

Direct logins

- Amazon.com
- American Airlines
- Bloglines
- del.icio.us
- Digg
- Expedia.com
- Flickr
- Google Account
- Google Calendar
- Google Docs
- Google Mail
- Jalku
- LinkedIn
- Lufthansa
- Microsoft Office CD Key
- MyBlogLog
- MySpace
- The New York Times
- Yahoo! Account
- Yahoo! Mail

Cards

American Airlines

Field label	Field data	Type
A-Advantage N.	M3215J2-1	text
Password	*****	text
Web site	http://www.aa.com	web address
Call center	1-800-222-2377	text
Expire date	24-11-2008	date

Direct logins

- American Airlines

edit

Checksums

Use the checksums below to make sure that Clipperz code hasn't been tampered with.

[More about using checksums](#)

- **Build:** 1346
- **Size:** 1.572.670 bytes
- **MD5:** 5b6b8 05e5 8ffe f2fd bf1e 0f99 d988 cfac
- **SHA1:** 91eb aaa6 8f97 3e8a 9d2d dd98 38c0 49bb 8364 99fd

Last update: 5 January 2009

It provides the code and the checksum of `index.html`, however checking for correctness, fairness, ... is not a trivial task! **Basically, the client has to trust the server**

Mash-up
threats

Mattia Monga

Web
applications

The old web

The rising of the
client side

Mash-ups

Highlights of
current
research

An open
(brave) new
world

Conclusions

What happen to my passwords?



DICo

Mash-up
threats

Mattia Monga

Web
applications

The old web

The rising of the
client side

Mash-ups

Highlights of
current
research

An open
(brave) new
world

Conclusions

- “Boundary” analyses doomed to failure due to encryption and covert channels;
- Signed scripts protect only from men in the middle and browser support is poor.

A mash-up of trust



DICo

Mash-up
threats

Mattia Monga

Web
applications

The old web
The rising of the
client side

Mash-ups

Highlights of
current
research

An open
(brave) new
world

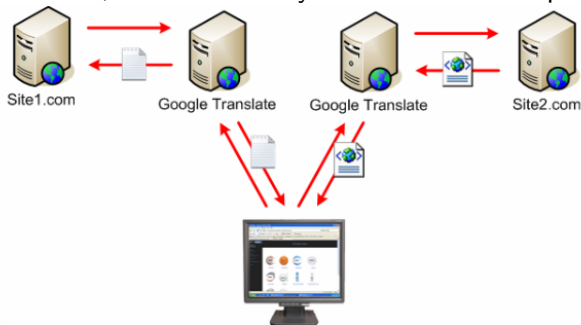
Conclusions

The problem is exacerbated by **mash-ups**: a web application that combines data or functionality from two or more sources into a single integrated application.

- Many principals to trust (sometimes even not easy to spot how many)
- Same origin policy voluntarily broken: interaction among different domains is a design goal!

Same origin policy in mash-up

Too strict, thus it is totally avoided! For example, with a proxy.



(from "JavaScript Malware for a Gray Goo Tomorrow!", by Billy Hoffman)

How one could possibly know if sensitive data leak from a (allowed) site to another (e.g., credit card number needed to book a hotel room goes to Google Maps)

Mash-up
threats

Mattia Monga

Web
applications

The old web
The rising of the
client side

Mash-ups

Highlights of
current
research

An open
(brave) new
world

Conclusions



- How to organize the data flow among different principals?
 - ① Automatic partition at building time (Chong et al., 2006)
 - ② Enhance the web infrastructure to document boundaries (De Keukelaere et al., 2008)
 - ③ Encapsulate mash sites in classical components with well defined interfaces (Baresi et al., 2009)

What are we working on?



DICo

Mash-up
threats

Mattia Monga

Web
applications

The old web
The rising of the
client side

Mash-ups

Highlights of
current
research

An open
(brave) new
world

Conclusions

Work in progress: **give to the client a way to monitor and/or enforce properties at run-time**. Approach based on aspect-oriented programming and static/dynamic analyses to match patterns.

- can leverage on parallel research on traditional applications, but web apps have specific issues:
 - multi-language more common
 - dynamic, reflective languages
 - client side computation added to achieve responsiveness. . . monitor overhead might be unacceptable



Eventually, the web seems to evolve to an **open world**. Are we prepared?

- context in which applications operate is (mostly) unknown at design-time
 - **How to specify and enforce security policies?**
- applications are assemblies of components discovered at run-time
 - **How one might possibly trust each other?**
- several stakeholders are involved in a service
 - **Is there a shared design (and protection) goal?**



- Protection based on a **centralized** reference monitor and a fixed sets of subjects and target objects is inadequate. (Dynamic proposals [UCON, Shandru 2003] still often assume centralized authorities trusted by anyone)
- How to assure a substantial **global fairness** with no (benevolent) dictators? How to measure the emerging properties of the system that can affect me, as a user of the system or provider of some of the services?

Game theoretical approaches might help to study emerging equilibria or devised proper mechanisms.



- The web is no more client-server: we need better tools to analyze/protect the client side
- The client side is growing fast in its complexity and danger: we need to empower (and educate) users
- Mash-ups (but also orchestrations or choreographies of web services) bring us an open and decentralized world of components: we need new conceptual models to approach security



Thank you!