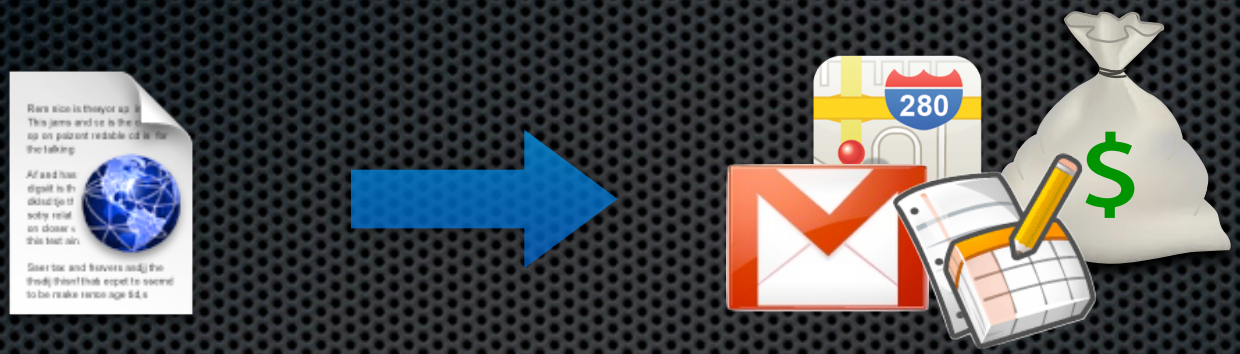
Supporting and Securing Programs inside Web Browsers

Charlie Reis

University of Washington / Google

Web is Evolving



Pages *Programs*

- More complex, active content
- Valuable data, targeted by attackers
- **Browser architectures need to support programs**

Outline

Current Browser Landscape

Security Challenges

Secure Site Isolation

Browser Wars Re-ignited

- **Many advances** in current & new browsers
- **Improving performance, features, robustness**
  - Security better, but still a big concern

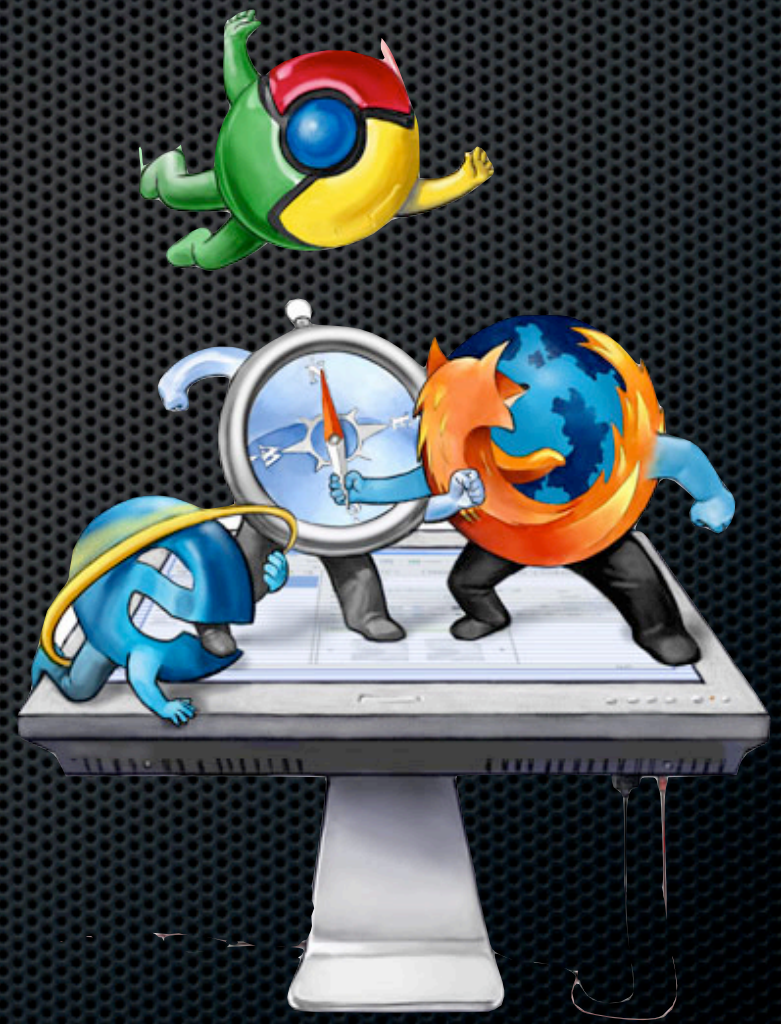
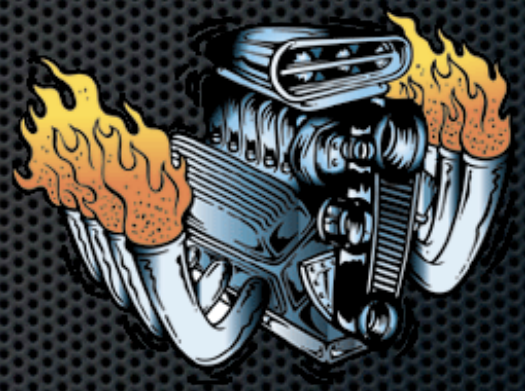


Image: Matt Collins / New York Times

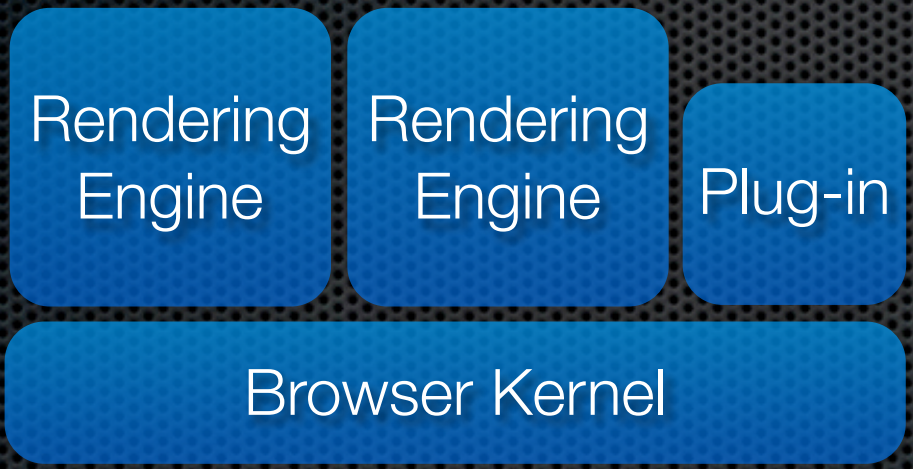
Performance & Features



- **Super-charged JavaScript engines**
  - Firefox 3.5, Chrome, Safari, Opera
  - Also memory reductions, native code execution
- **HTML5, Gears, Browser Plug-ins**
  - Offline, storage, workers, device access
  - *New surface area for attacks*

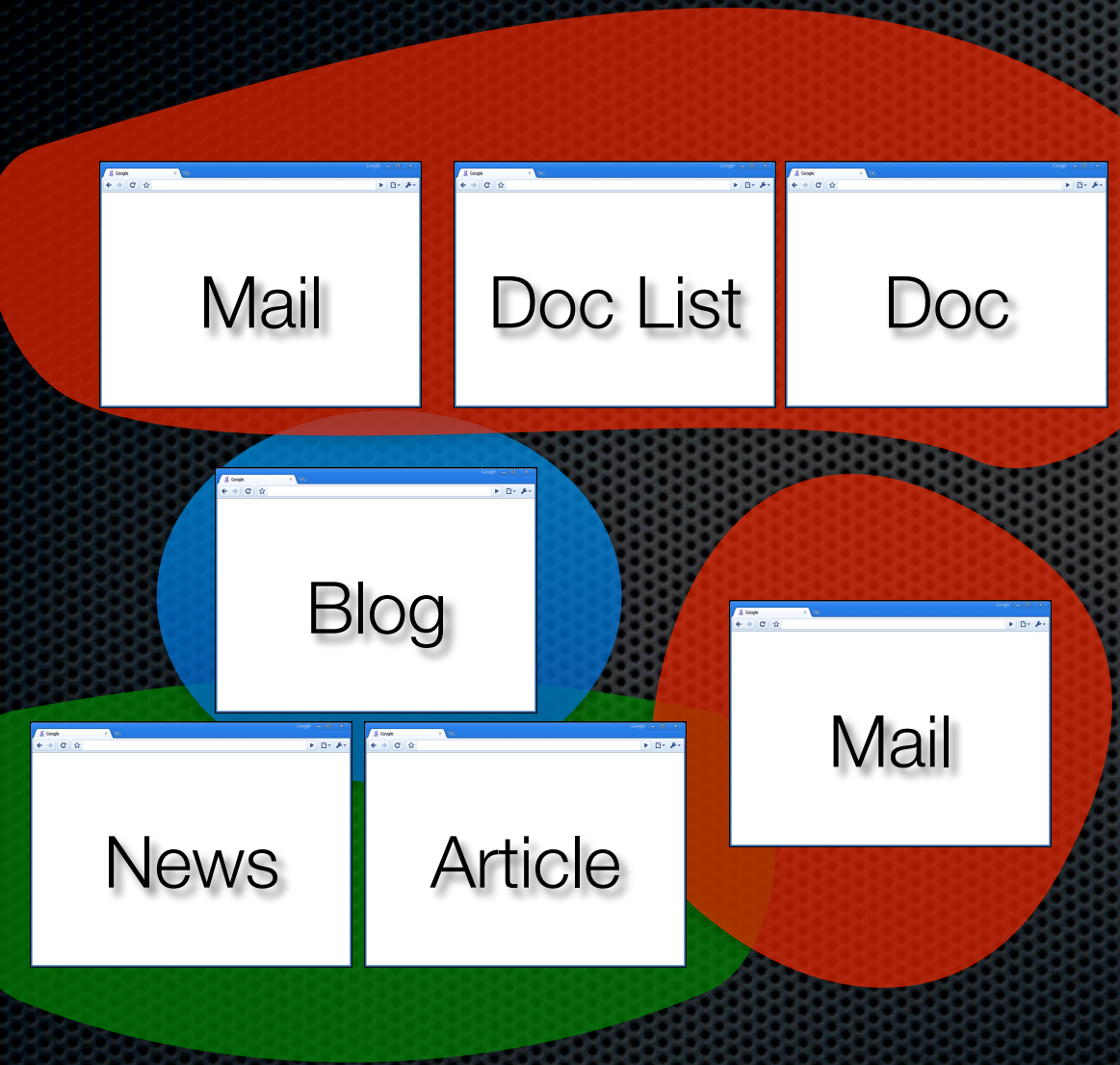
Robustness

- **Multi-process architectures** (Google Chrome, IE8)



- **Program abstractions**
  - Site Instances (Google Chrome)

Site Instances



- Set of same-site pages that share references
  - Safe to isolate with OS processes
  - Compatible program abstraction

Outline

Current Browser Landscape

Security Challenges

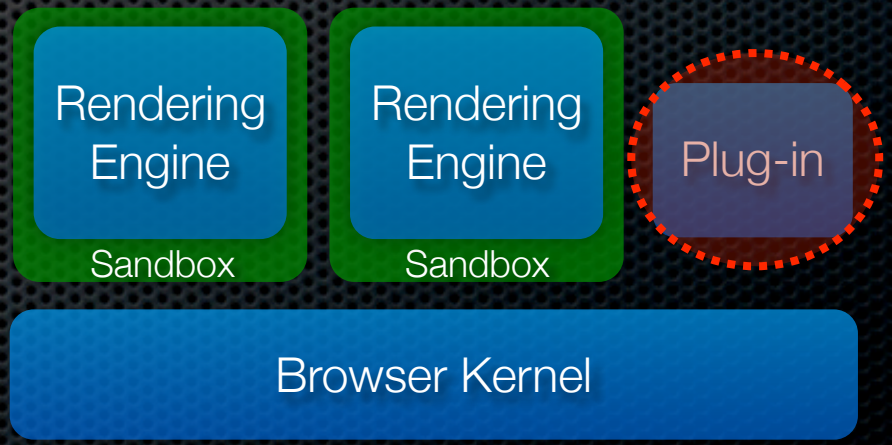
Secure Site Isolation

Improving Security

- How can browser's architecture help?
  1. Protect user's **local resources**
     (Seeing progress in real browsers)
  2. Protect user's **web principals** from each other
     (Challenges in practice)
  3. Protect user's and publisher's **intentions**
     (Research progress)

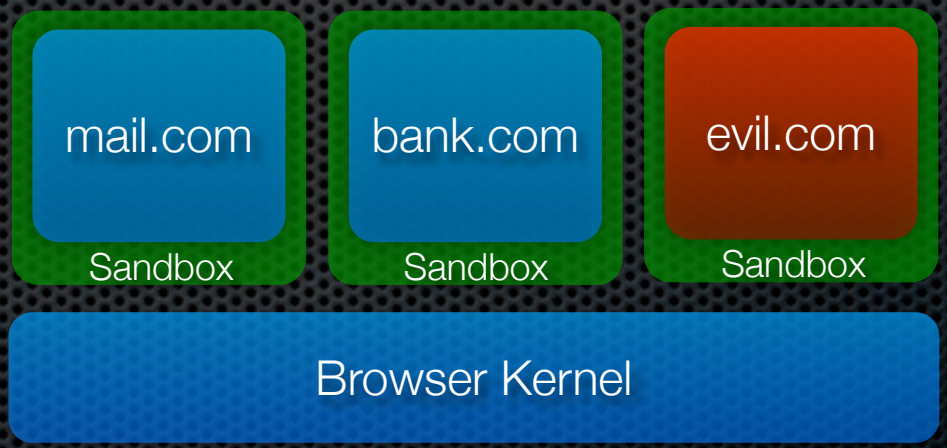
1. Protect Local Resources

- **Limit damage to client,** despite exploits
  - Run web apps with low privileges
    - **Low rights IE:** renderer can't write to disk
    - **Chrome's sandbox:** renderer can't access local resources
- Plug-ins still a concern...



2. Protect Web Principals

- Can we protect user's web accounts despite exploits?
  - Not as simple, if **compatibility** is important…
    (will return to this)



3. Protect Intentions

- **User's intentions**
  - Prevent UI redressing *(David's talk)*
- **Publisher's intentions**
  - Anti-XSS mechanisms *(e.g., BEEP)*
  - Detect in-flight changes *(e.g., Web Tripwires)*

Outline

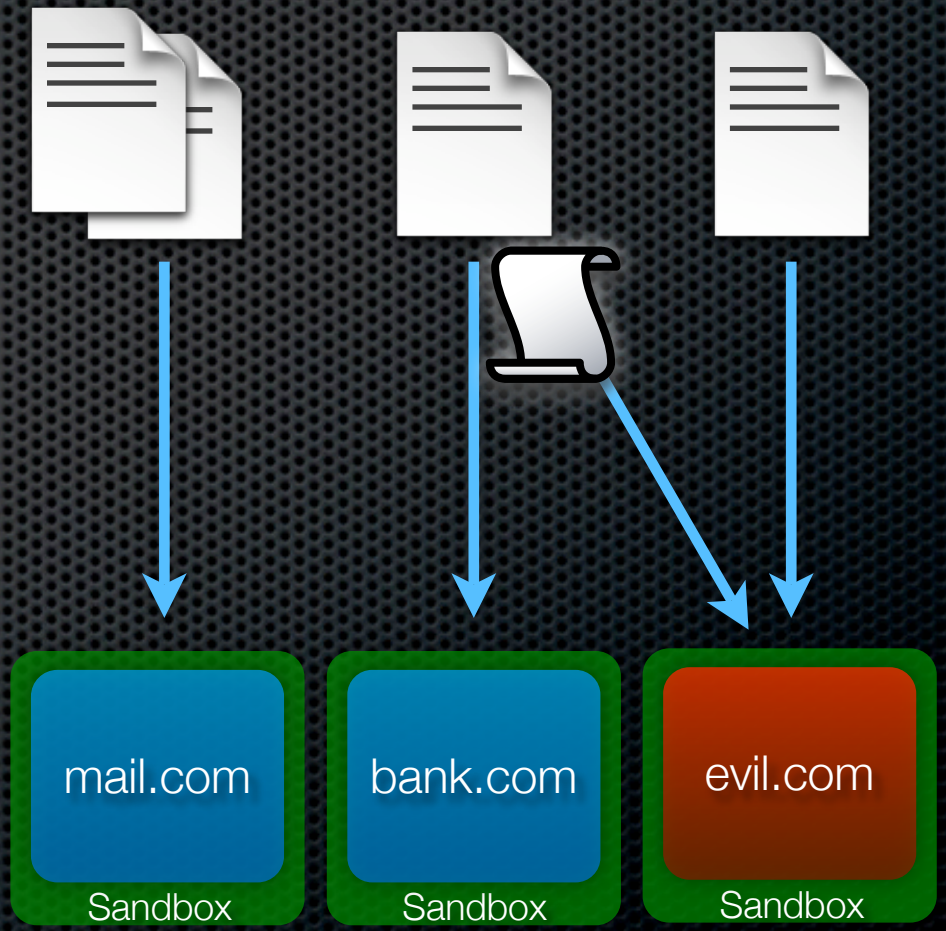
Current Browser Landscape

Security Challenges

Secure Site Isolation

Trouble with Sub-resources

- Could always isolate pages **based on site**
- Pages can load objects from any site
  - Requested with **user's credentials**
  - Contain private info!



Example: Gmail Contacts

(Grossman)

- **Evil site loads JS file from Gmail, containing contacts**
  - Intended for XHR; run by evil site
  - **Past:** CSRF vulnerability, leaks info
  - **Present:** add “while(1)” to script
    - Prevents leak, as long as renderer’s logic is correct

Relying on Renderer

- Embedded objects must be "**opaque**"
  - Scripts are execute-only
  - Images, etc., can't be sent back to server
  - **Enforced by logic inside the renderer**

- Can we protect user's other accounts, **even if a renderer is exploited**?

Potential Solutions

Alternative World: SSBs

- Imagine using a separate browser for each site
  - e.g., **Site Specific Browsers** (Prism, Fluid)
- Each has its **own set of credentials**
  - Can’t be abused by other sites in different browsers

Credential Isolation

- **Apply same idea in a single browser?**
  - Each site gets its own cookie store, etc.
  - No cross-site cookies sent on sub-resources
- **Goal:** Site Instance never contains data it can't access

Drawbacks

- **Not all credentials are explicit** (e.g., IP address)
- **Breaks sites** that depend on cross-site cookies
  - e.g., Verisign PIP, Facebook?, Advertisers?
- **What does following a cross-site link do?**
  - *Safe?* (Send the cookie and stay logged in?)
  - *Unsafe?* (CSRF attack attempt?)

(RequestRodeo)

Alternative Approaches

- **Distinguish types of cookies?**
  - Per-instance vs Browser-global?
  - Like CSRF tokens within the browser
- **Origin headers on all sub-resources?**
  - Let server decide whether to send data
  - Privacy concerns...

Questions and Discussion

- **Are we facing a fundamental decision?**
  - Open mashups vs walled applications?
  - Or just a need for new mechanisms?
- How to **compatibly + securely** isolate sites?
- How to **sandbox plug-ins**?