Monitor distribution

No Author Given

No Institute Given

Abstract. Various implementations of ways to distribute a monitor to the clients.

1 Introduction

2 Mashups

Describe mashups, difficulties in mashup security and information flows in mashups.

3 Architecture

Describe various monitor distribution techniques.

3.1 Shadowing

Inline the monitor into the JavaScript code. Appealing idea, previous work for toy languages, but difficult in practice. Abandoned due to complexity with coercion in expressions.

3.2 Plugin

Plugin replaces JavaScript interpretation. Install plugin and you're done. - Browser dependent - Intrusive to the browser - Cumbersome for user to setup/configure - General, every browsed page

3.3 Proxy

Proxy injects monitor into every page and wraps all JavaScript. Setup the proxy and you're done. - Practically the same monitor as the plugin - Browser independent - Less intrusive to the browser, simple proxy setup - Less cumbersome to setup - Server-side configuration - General, every browsed page

3.4 Service

Service injects browsed pages with the monitor and references to itself, and wraps all JavaScript. Browse through service and you're done. - More intrusive in HTML content, rewrite links and external references - Browser independent - Not intrusive to the browser - Even less cumbersome to setup, browse to server URL - Server-side and client-side configuration possible - Less general, only browsed pages

3.5 Integrator

The mashup integrator includes the monitor and decides which JavaScript should run within the monitor. (Don't mention sandbox.) You're done. - No setup - Not intrusive to the browser - Not general - Deliberately intrusive to HTML - Integrator configuration, no user control - Run independent part of the code outside the monitor, possible performance gains

4 Implementation

Describe features and drawbacks with each implementation.

4.1 Plugin

- rewrite all or some, client-side configuration - innerHTML - difficult to mimic execution order in the page parsing - document.write - firefox challenges - execution order (script inclusion heavy but possible) - execution contexts no problem (SVG and CSS uses native JavaScript)

4.2 Proxy

- rewrite all or nothing, problem with passing configuration to proxy - DOM parsing to find scripts, compared to regular expressions - difficult to identify all contexts (SVG, CSS and PDF can be rewritten, not flash)

4.3 Service

- rewrite all or some, can pass configuration through cookie - piggybacking, wildcard DNS compared to URL parameter - disable SOP, domain relaxing - DOM parsing to find scripts, compared to regular expressions - difficult to identify all contexts (SVG, CSS and PDF can be rewritten, not flash)

4.4 Integrator

- rewrite some, integrator driven, no or limited user configuration - no need for parsing, integrator decides - well defined per site policy - site could allow user to configure policy - requires developer understanding of the monitor - future reflection of monitored code?

5 Case study

- mlcalc.com - newyorker.com - mockup vs. real deal

6 Related work

7 Conclusions

References