JavaScript Security: Let's Fix It

Brendan Eich
CTO, Mozilla Corporation

(We could bundle it)

bugs to fix

- <script> injection: perl.com ⇒ pr0n.com

  (http://radar.oreilly.com/2008/01/dangers-of-remote-javascript.html)

- lure.org has <form action=bank.com/...>
  and <script>document.forms[0].submit()

- mashup.com has to trust maps.google.com
  if it loads <script src=maps.google.com>

helpful, yet not enough

- ECMAScript, 5th Edition (ES5)
- postMessage (IE8, Firefox 3, Safari 4, etc.)
- http://code.google.com/p/google-caja/
- http://websandbox.livelabs.com
- ADsafe, Jacaranda, other verifiers

why not enough?

- "Advisory" -- no mandatory enforcement
- No change to lax default cross-site policies
- Variously complex
- Tyranny of choice
- Programmers always cut corners

FlowSafe

- A Mozilla project (WebKit, chromium next)
- Academic/industrial/open-source collaboration
- Prof. Cormac Flanagan, UC Santa Cruz
- Prof. Michael Franz, UC Irvine
- Dr. Andreas Gal, Brendan Eich, Mozilla

challenge

- Integrity is not enough: web developers need better confidentiality properties
- Label pc, addresses, and all values
- While not losing the JS performance wars
- Improve the browser's default security policy beyond SOP using information flow
- Without static analysis for implicit flows

key ideas

- Monitor **all** references
- Efficient sparse labeling
- Fail-stop “no-sensitive-upgrade” check to preserve non-interference
- Trace-JIT fast path optimizations

implicit flow

- Given a secret in x:
  - y = true;
  - z = true;
  - if (x) y = false; // taint y
  - if (y) z = false; // not z
- Implicit flow from x to z

no-sensitive-upgrade

- Assignment to variable y must fail-stop if original label of y $\subset$ pc (label of x)
- Principle: code conditioned by secret (x) can't upgrade a non-secret (y)
- Script may call upgrade(y) before if (x) ... to continue rather than fail-stop
- Leak "half a bit" in the x == false case

sparse labeling

- A value v is either unlabeled raw value r
- Or else a pair r^k of raw value r and label k
- Label with respect to implicit label pc is
  - $label_{pc}(r) = pc$
  - $label_{pc}(r^k) = pc \cup k$
- Semantic rules split into fast, slow paths

more sparse labeling

- Implicit label pc applies to same-origin code and data; other-origin gets explicit label
- Implementation: implicit label per GC page for fast access and low space overhead
- Explicit label requires a transparent box or lightweight wrapper

fast vs. slow path

- Constants and local variables are unlabeled
- Calling unlabeled closure entails no labeling
- Calling labeled closure labels return value
- For var x = r, leave label pc on r implicit if label(x) = pc
- For var x = r^k, enforce no-sensitive-upgrade and pass only if pc ∪ k ⊆ label(x)

results so far

- Big-step operational semantics
- Correctness and non-interference proofs
- SML implementations for unlabeled, sparse, and universal labels
- Unlabeled / sparse / universal: 1 / 1.2 / 1.7

policy ideas

- Prevent $r^{k \cup pc}$ from flowing to *any* server with where eTLD+1(k) != eTLD+1(pc)
- Save perl.com: label <script src="...ad.js"> with k("...ad.js") ∪ pc, restrict DOM access, geometry, z-order, location.href = ...
- Markup isolation + label tags = secure distributed mashups, GreaseMonkey, etc.

issues

- Is fail-stop usable? may need conservative/ approximate static analysis
- Explicit labels must round-trip through rendering/presentation back up to DOM (e.g., :visited tracking)
- Timing, half-a-bit, other information leaks
- Foolproof sanitize(v)...

comments welcome

- cormac@ucsc.edu
- franz@uci.edu
- gal@mozilla.com
- brendan@mozilla.com