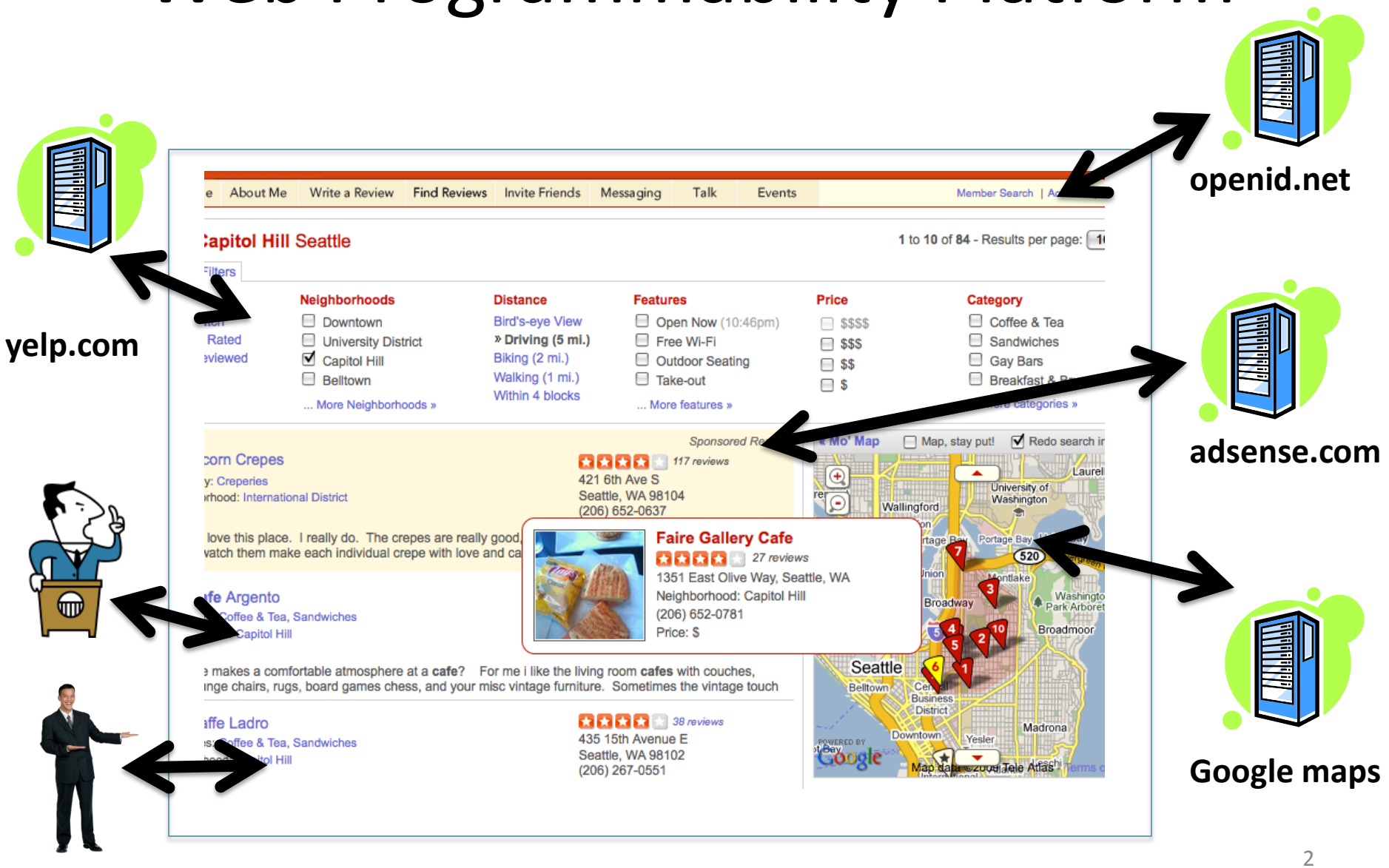
ConScript

Specifying and Enforcing Fine-Grained Security Policies for JavaScript in the Browser

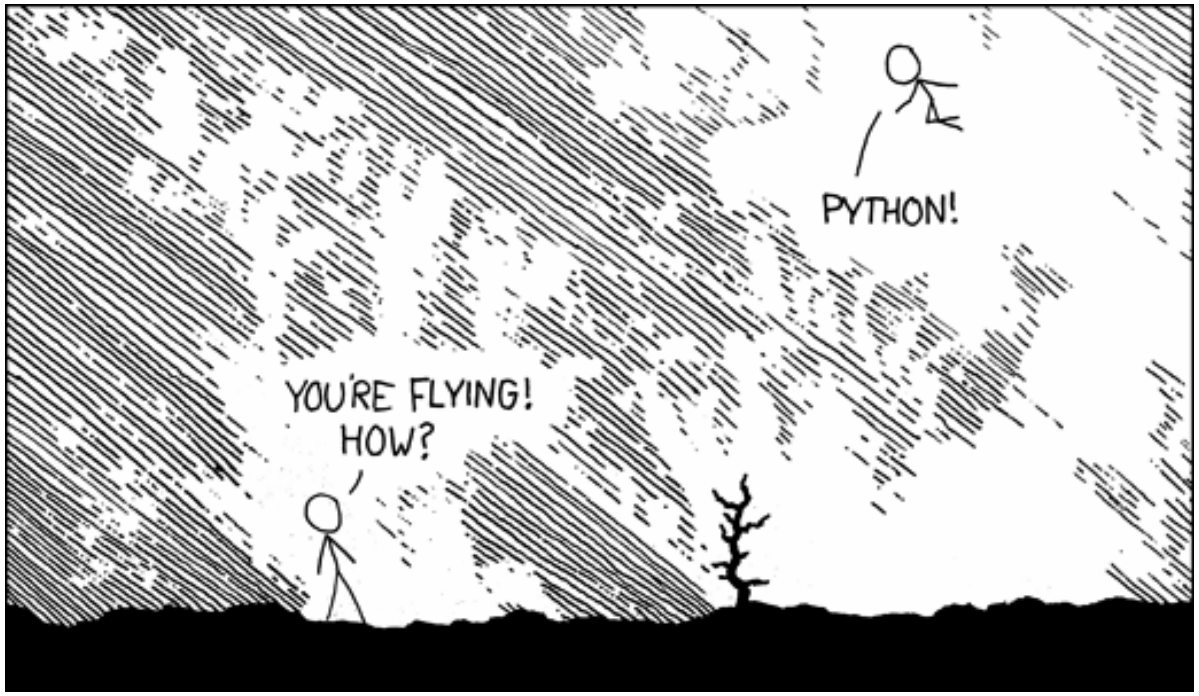
Leo Meyerovich
UC Berkeley

Benjamin Livshits
Microsoft Research

Web Programmability Platform



Rich Internet Applications are Dynamic



Yelp.com:

main.js

… jQuery.js

… adSense.js

… GoogleMaps.js

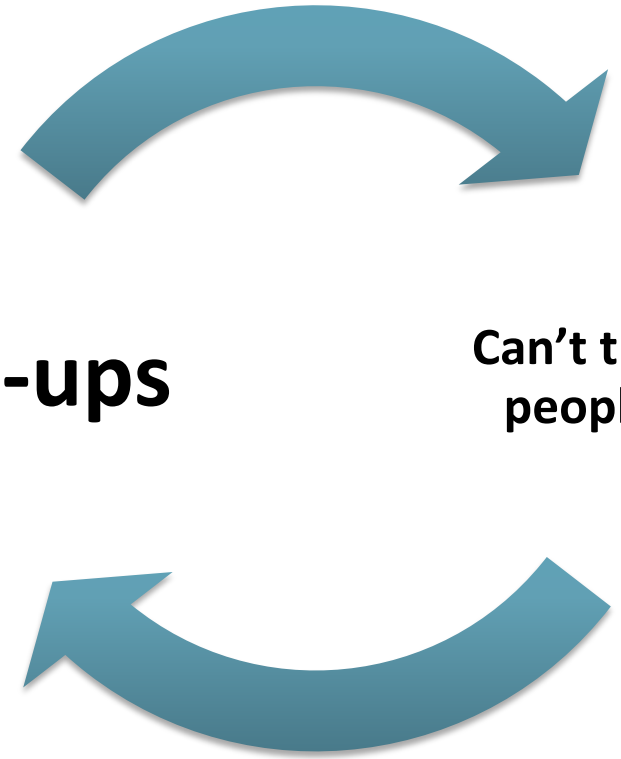
… OpenID_API.js

flexible runtime composition

… but little control.

Towards Safe Programmability for the Web

Mash-ups



Can't trust other people's code

Goals and Contributions

control loading and use of scripts	• protect benign users • by giving control to hosting site • ConScript approach: aspects for security
express many policies *safely*	• 17 hand-written policies • correct policies are hard to write • proposed type system to catch common attacks • implemented 2 policy generators
browser support	• built into IE 8 JavaScript interpreter • runtime and space overheads under 1% (vs. 30-550%) • smaller trusted computing base (TCB)

approach
protect benign users by giving control
to the hosting site
: aspects for security

ConScript

- Approach
  - protect benign Web users
  - give control to the hosting site
- How
  - Browser-supported aspects for security

Contributions of ConScript

A case for aspects in browser

- protect benign users by giving control to hosting site
- ConScript approach: aspects for security
- built into IE 8 JavaScript interpreter

Correctness checking

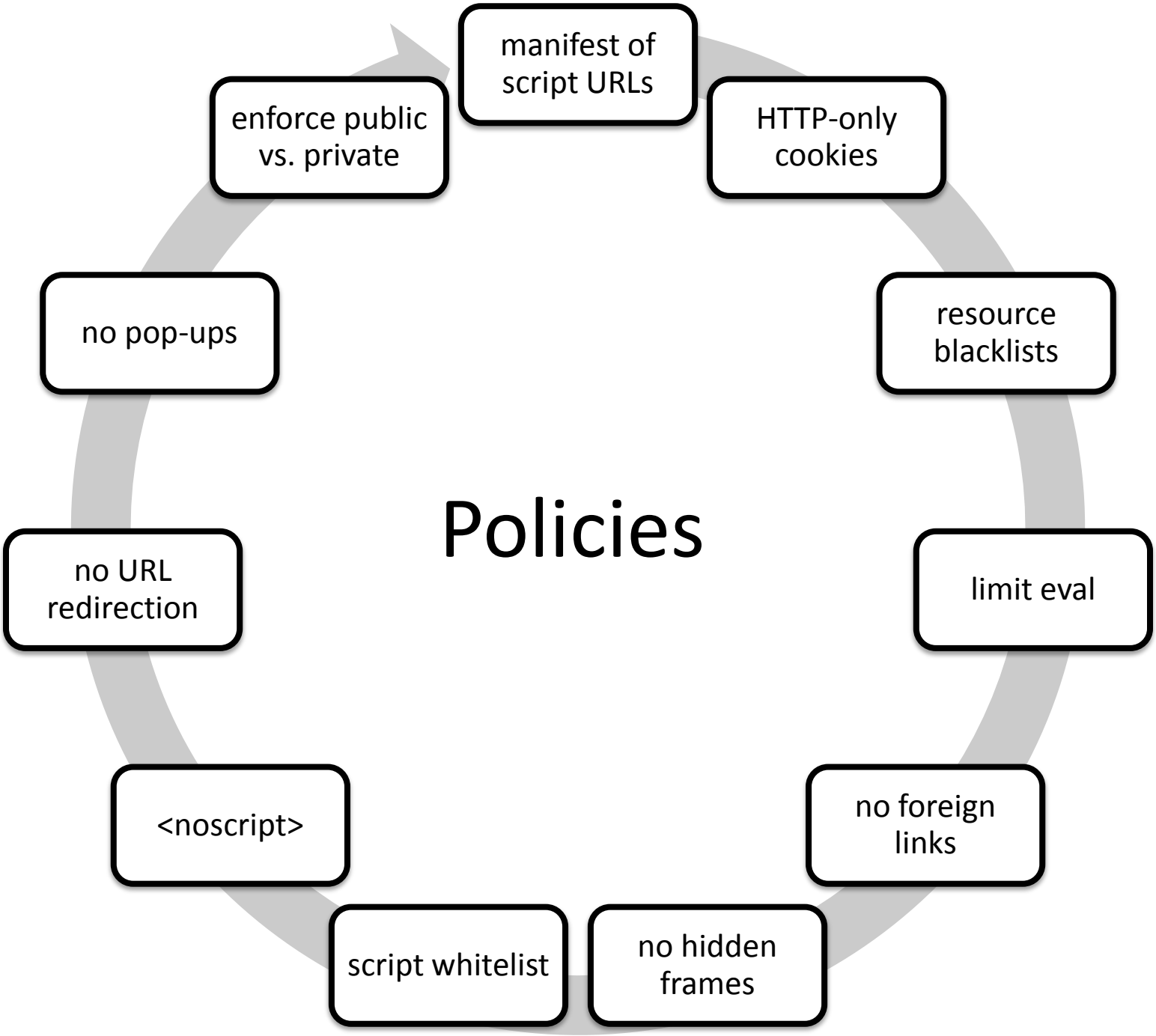
- Policies are easy to get wrong
- Type system to ensure policy correctness

Expressiveness

- 17 hand-written policies
- Comprehensive catalog of policies from literature and practice
- implemented 2 policy generators

Evaluation

- Tested on real apps: Google Maps, Live Desktop, etc.
- runtime and space overheads under 1% (vs. 30-550%)
- smaller trusted computing base (TCB)


Policies
manifest of script URLs
HTTP-only cookies
resource blacklists
limit eval
no foreign links
no hidden frames
script whitelist
<noscript>
no URL redirection
no pop-ups
enforce public vs. private

ConScript aspects

implementing aspects in IE8

checking ConScript policies

generating ConScript policies

performance

No **postMessage**: A Simple Policy?

Wrapping: [[Caja, DoCoMo, AOJS, lightweightjs, Web Sandbox, …]]

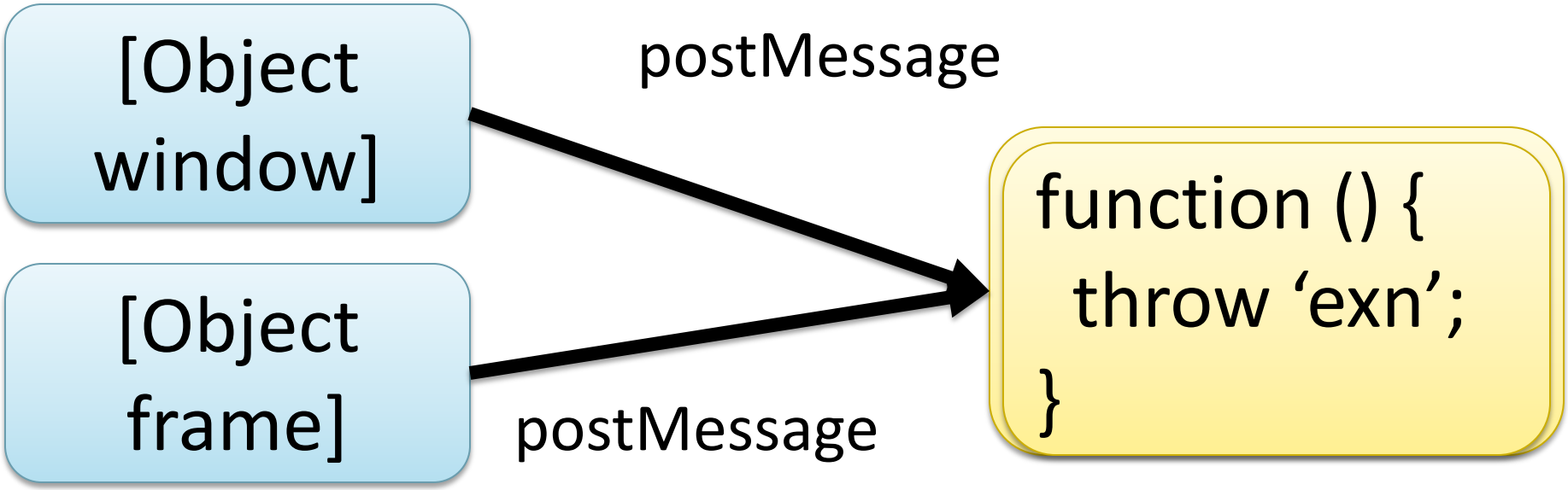
```
window.postMessage = function () {};
frame1.postMessage("msg", "evil.com")
```

Aspects: [[AspectJ]]

```
void around(String msg, String uri) :
call DOM.postMessage(String m, String u)
{   /* do nothing instead of call */   }
```

… no classes in JavaScript / DOM …

Specifying Calls using References




```
around(window.postMessage,
       function () { throw 'exn'; });
```

ConScript Interface

1. Functions

   DOM: `aroundExt(postMessage, function (pm2, m, uri) { … });`

   JS: `aroundNat(eval, function (eval, str) { … });`

   User-defined: `aroundFnc(foo, function (foo2, arg1) { … });`

2. Script introduction

   <script>: `aroundScr(function (src) { return src + ';' + pol;});`

   inline: `aroundInl(function (src) { return src + ';' + pol;});`

ConScript aspects

implementing aspects in IE8

checking ConScript policies

generating ConScript policies

performance

Problem: Implementation?

Source Rewriting [[aojs, docomo, caja, sandbox, fbjs]]

```
function f () { … }
```

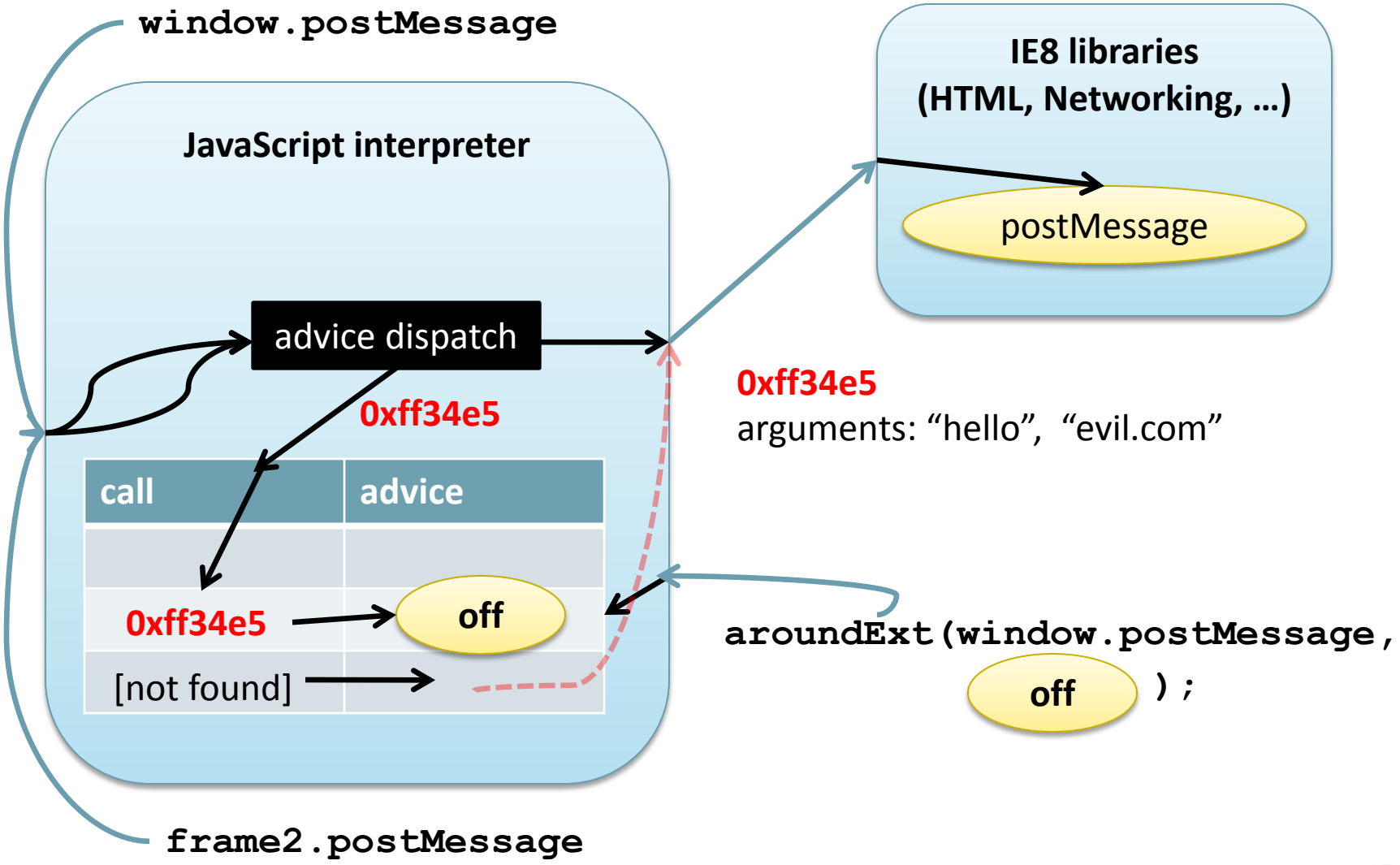
```
function f () {<before> … <after>}
```

☹ 50%-450% more to transfer, 30-70% slowdown

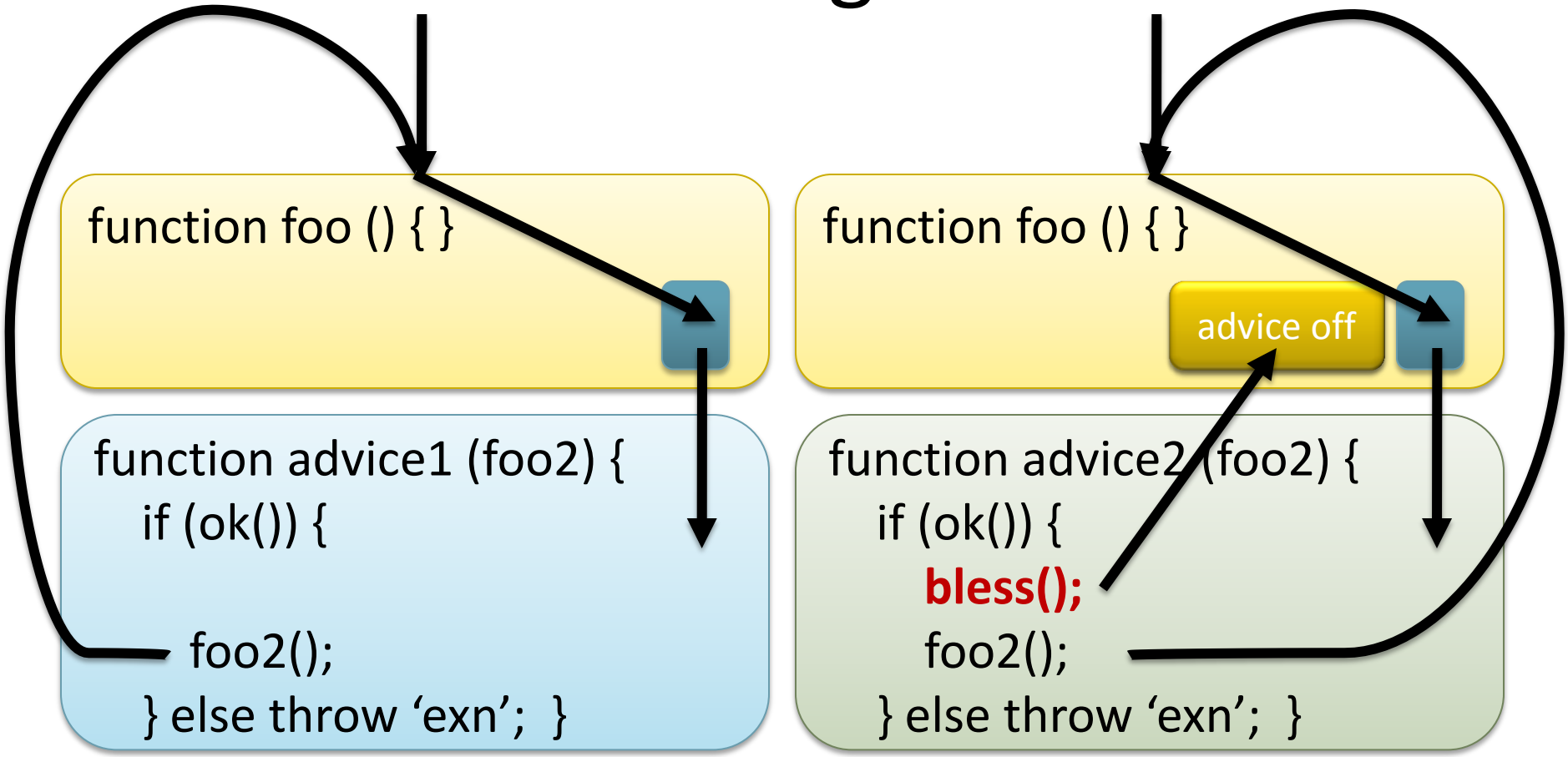
☹ limited: native (DOM) functions, dynamic code?

☹ big assumptions: adds parser to TCB, …

Mediating DOM Functions

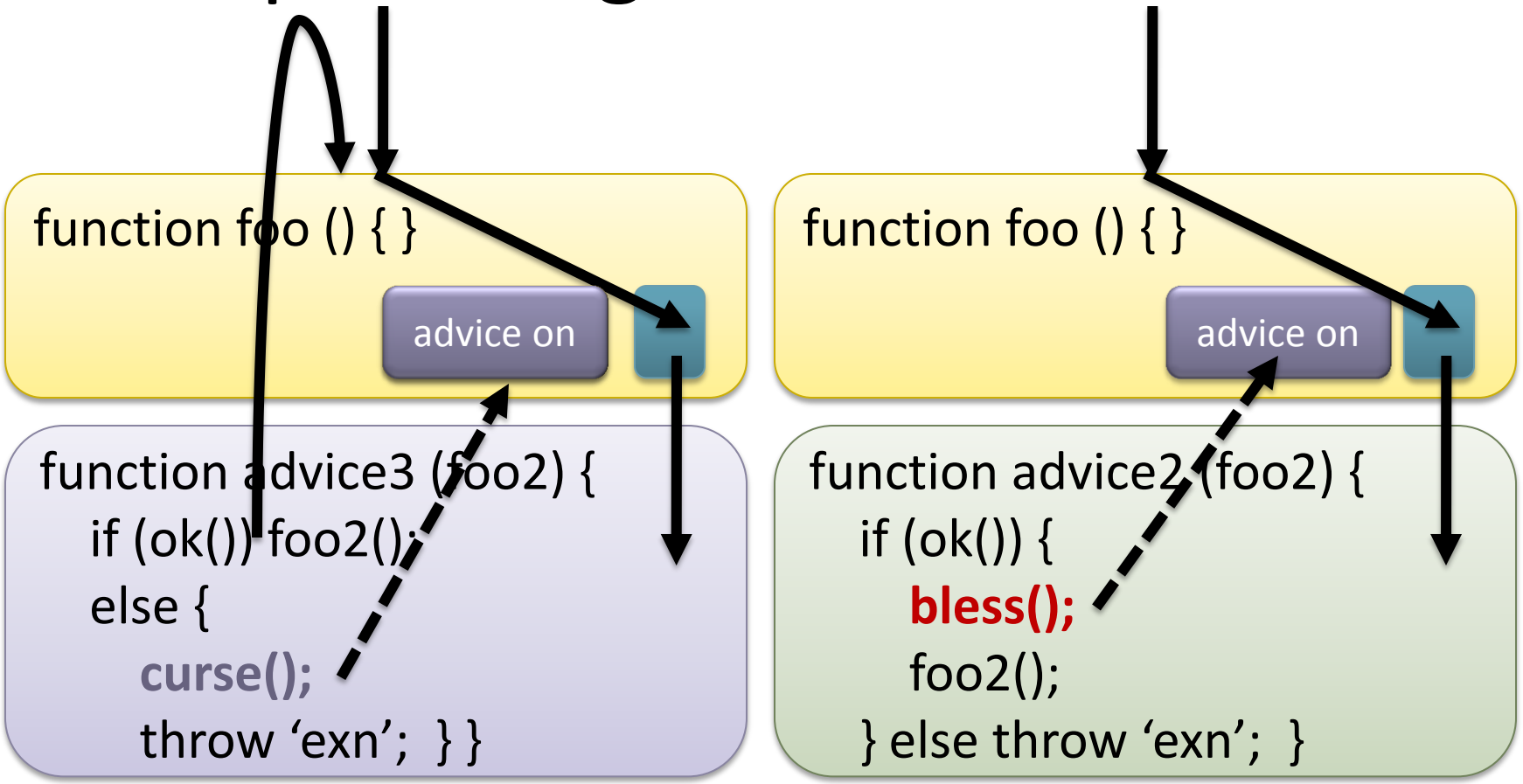


Resuming Calls



bless() temporarily disables advice for next call

Optimizing the Critical Path



- calling advice turns advice off for next call
- **curse()** enables advice for next call

ConScript aspects

implementing aspects in IE8

checking ConScript policies

generating ConScript policies

performance

Basic Usage

script whitelist

Yel

..

..

..

..

SURGEON GENERAL'S WARNING

Policies are written in a small JavaScript subset.

Applications only lose a few dangerous features.

```
<script src="main.js" policy="noEval()"/>
```

Policy Integrity

Objects defined with policy constructors do not flow out

Old Policy

```
around(postMessage, function (m, url) {
  w = {"msn.com": true};
  …
```

Policy Integrity

Objects defined with policy constructors do not flow out

Old Policy

```
around(postMessage, function (m, url) {
  w = {"msn.com": true};
  …
```

policy object: must protect

unknown: do not pass privileged objects!

Policy Integrity

Objects defined with policy constructors do not flow out

Old Policy

```
around(postMessage, function (m, url) {
  w = {"msn.com": true};
  …
```

User Exploit

```
postMessage("", "msn.com");
w["evil.com"] = 1;
postMessage("", "evil.com");
```

Policy Integrity

Objects defined with policy constructors do not flow out

New Policy

```
around(postMessage, function (m, url) {
  var w = {"msn.com": true};
  …
```

User Exploit

```
postMessage("", "msn.com");
w["evil.com"] = 1;
postMessage("", "evil.com");
```

Policy Integrity

Objects defined with policy constructors do not flow out

New Policy

```
around(postMessage, function (m, url) {
    var w = {"msn.com": true};
    …
```

policy object: must protect

unknown: do not pass privileged objects!

Maintaining Integrity

1. **Policy objects do not leak out of policies**
2. **Access path integrity of calls** (no prototype hijacking)

- ML-style type inference
  - ☹ basic
  - ☺ program unmodified
  - 😐 only manually tested on policies
- JavaScript interpreter support
  - call(ctx, fnc, arg1, …), hasOwnProperty(obj, “fld”)
  - caller

Transparency

- If running with policies throws no errors
  - … for same input, running without should be safe
  - **empty advice should not be functionally detectable**
- Difficult with wrapping or rewriting
  - `Function.prototype.apply, exn.stacktrace, myFunction.callee, arguments.caller, myFunction.toString, Function.prototype.call`
  - correctness vs. compatibility vs. performance …
- Simpler at interpreter level
  - rest up to developer
  - no proof

ConScript aspects

implementing aspects in IE8

checking ConScript policies

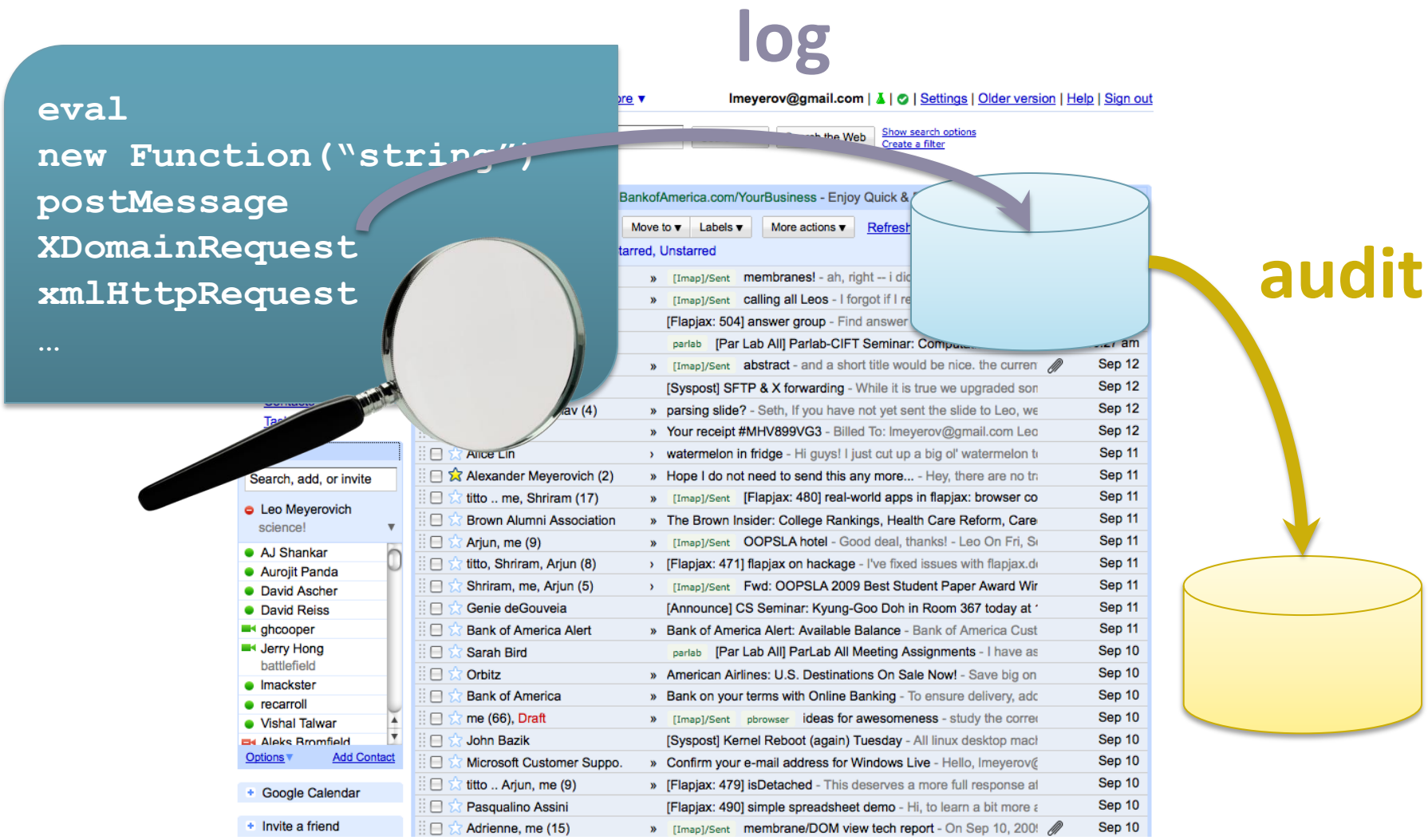
generating ConScript policies

performance

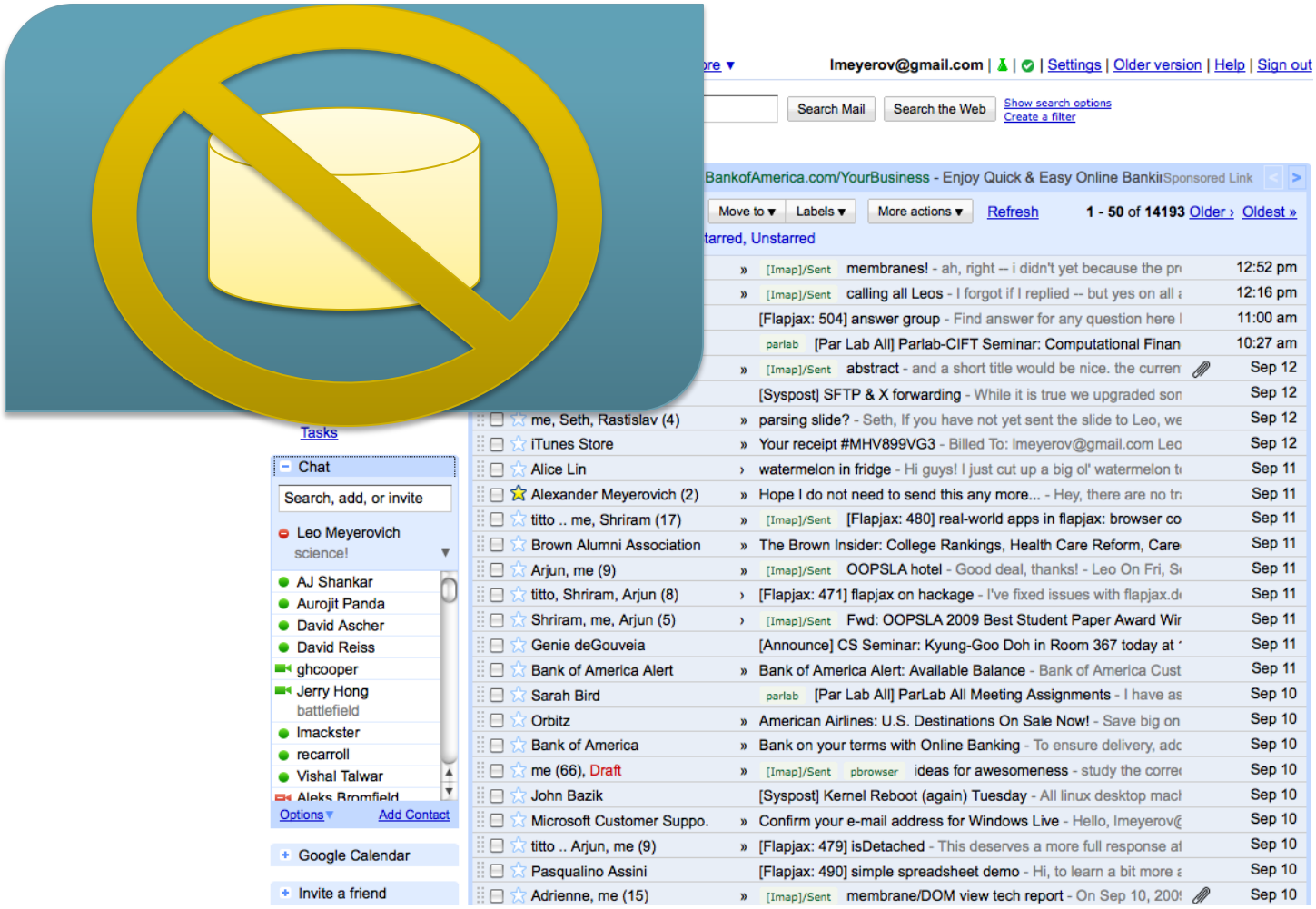
Automatically Generating Policies

- Intrusion detection
  - can we infer and disable unneeded DOM functions?
- C# access modifiers
  - can we enforce access modifiers like *private*?
- ASP policies
  - can we guarantee no scripts get run in <% echo %>?

Intrusion Detection 1: Learn Blacklist

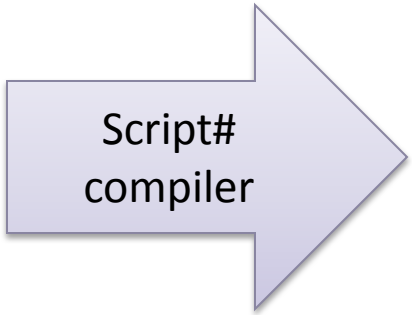


Intrusion Detection 2: Enforce Blacklist



Enforcing C# Access Modifiers

```
class File {
    public File () { … }
    private open ()  { … }
    …
```




```
function File () { … }
File.construct = …
File.open = …
…
```

C#

JavaScript


```
around(File, pubEntryPoint);
around(File.construct, pubEntryPoint);
around(File.open, privCall);
```

ConScript

ConScript aspects

implementing aspects in IE8

checking ConScript policies

generating ConScript policies

performance

Performance

Microbenchmarks: 1.2x (vs. 3.4x)

Initialization time: 0-1%

Runtime: 0-7% (vs. 30+%)

File size blowup: < 1% (vs. 50+%)

Microbenchmark: Mediation Overhead

```
var raw = obj.f;
obj.f = function () { raw();}
```

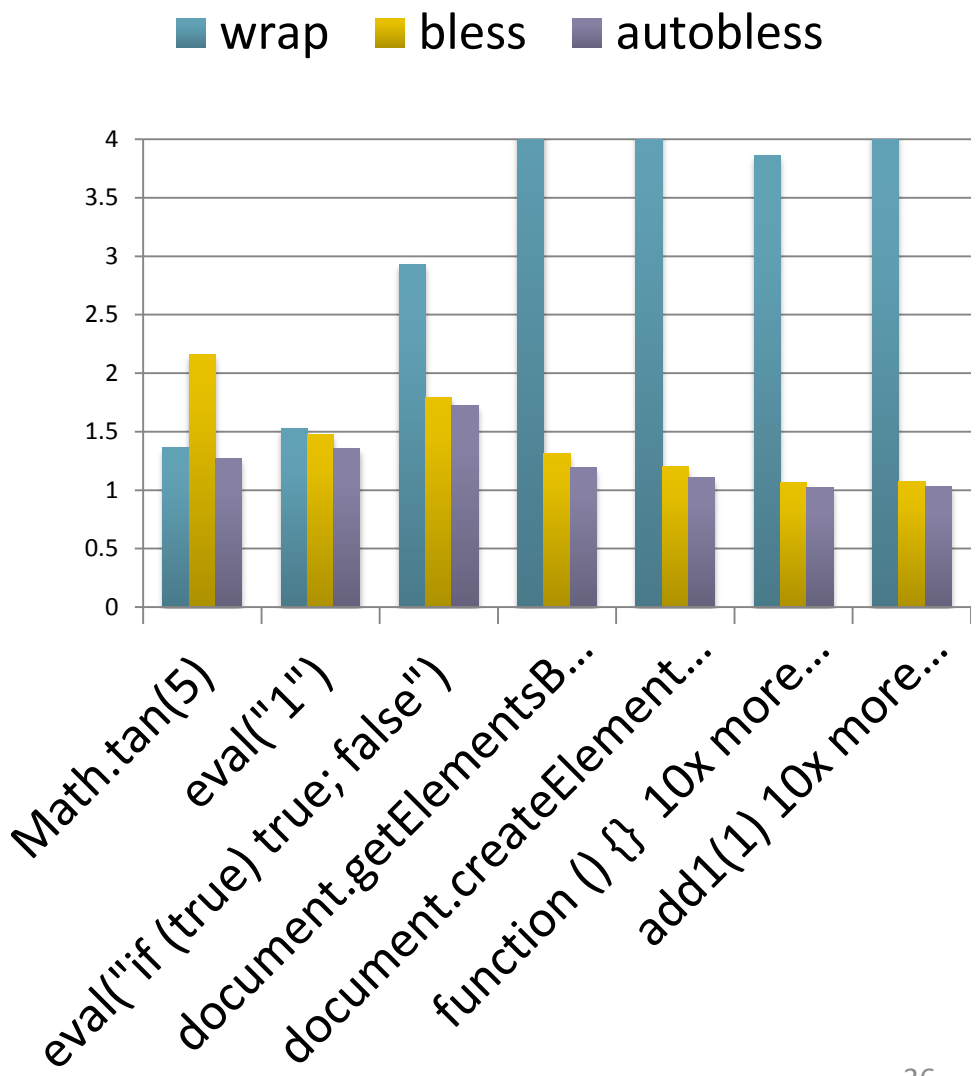
3.42x

```
function advice2() {
  bless();
  foo2();
}
```

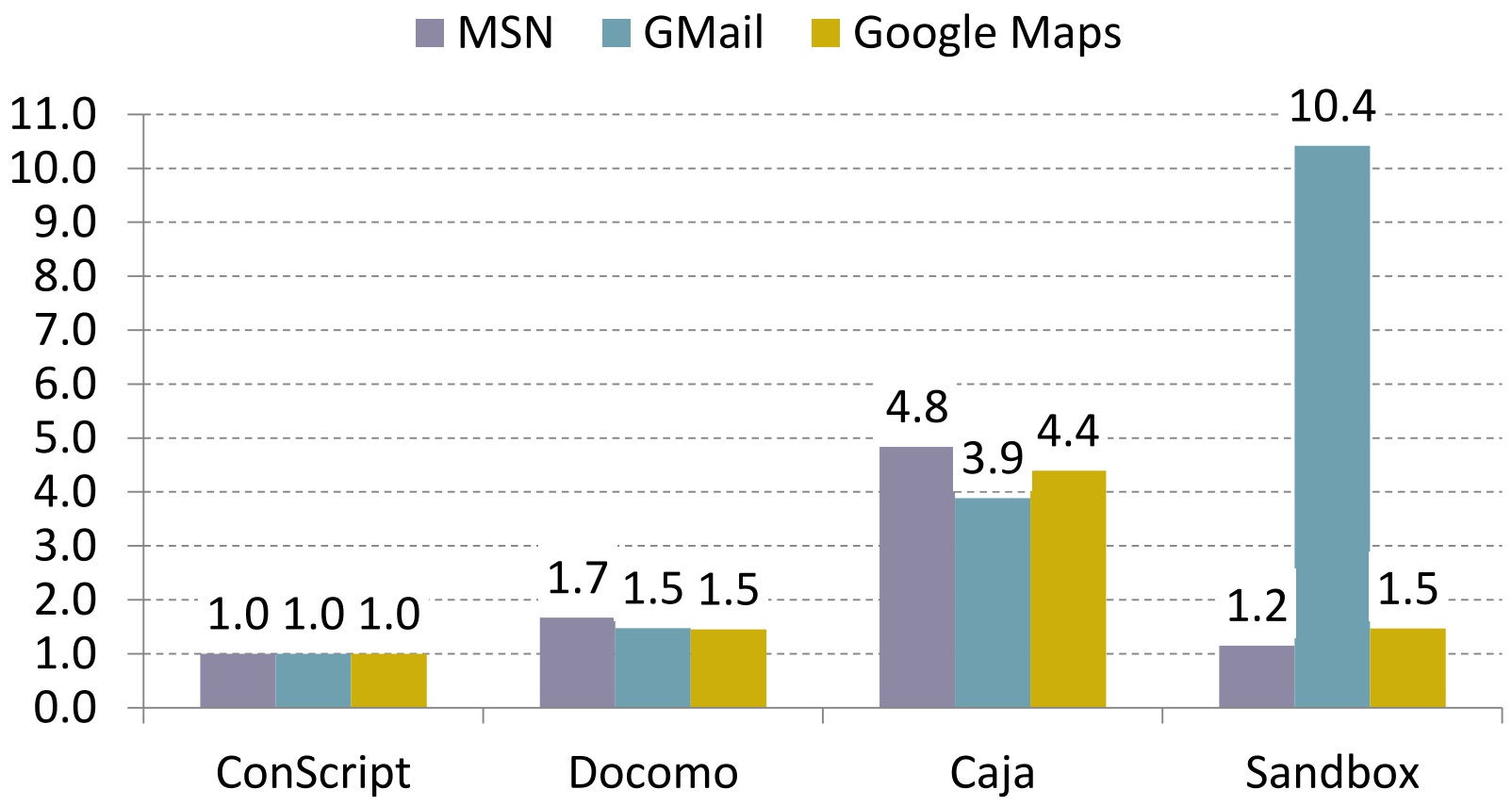
1.44x

```
function advice3() {
    foo2();
}
```

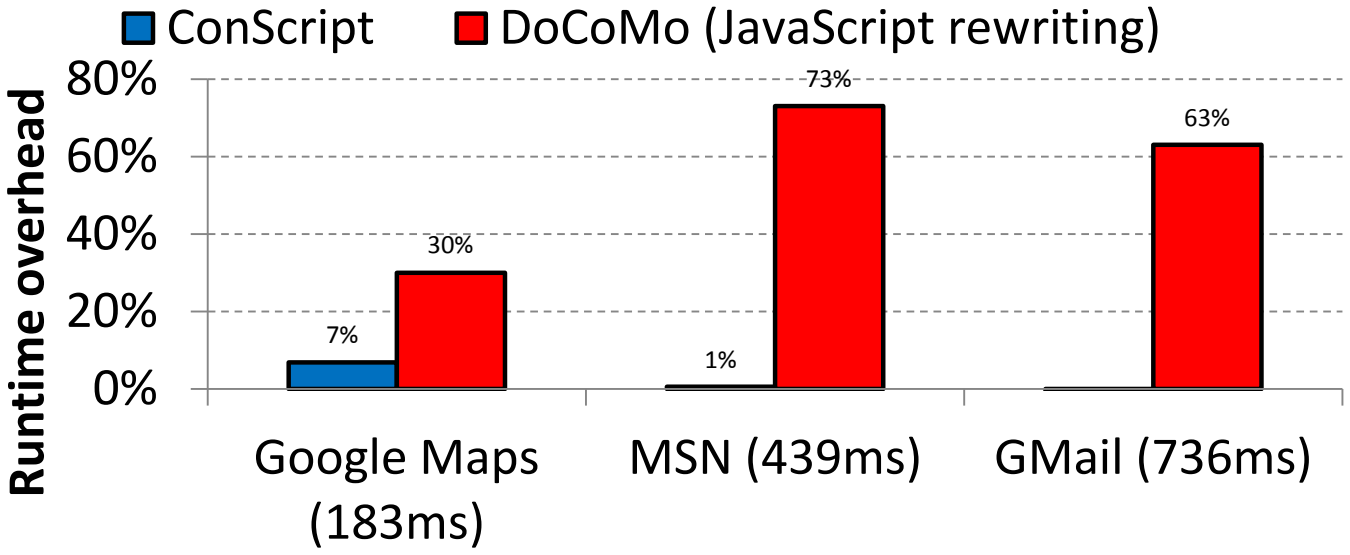
1.24x



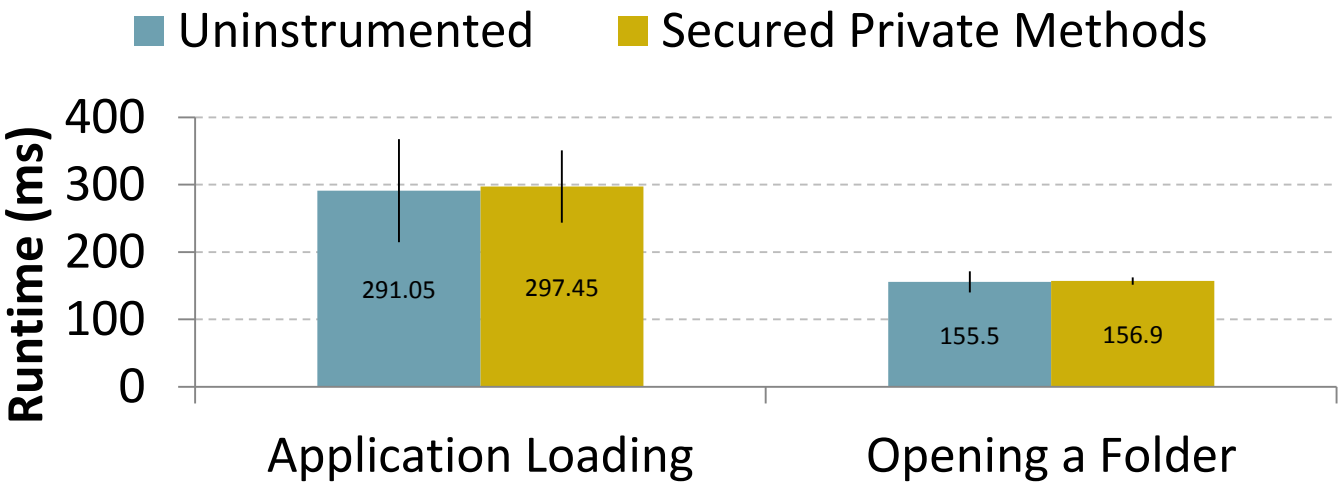
File Size Increase (IDS)



Runtime Overhead



Intrusion Detection System



Access Modifier Enforcement

Goals and Contributions

control loading and use of scripts

- protect benign users
- by giving control to hosting site
- ConScript approach: aspects for security

express many policies *safely*

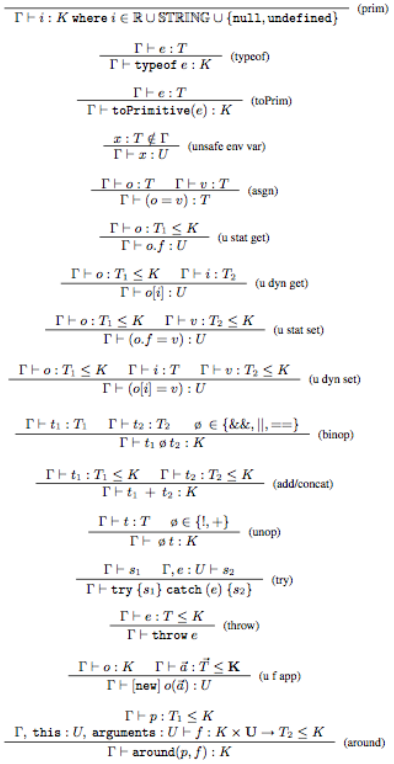
- 16 hand-written policies
- correct policies are hard to write
  - proposed type system to catch common attacks
  - implemented 2 policy generators

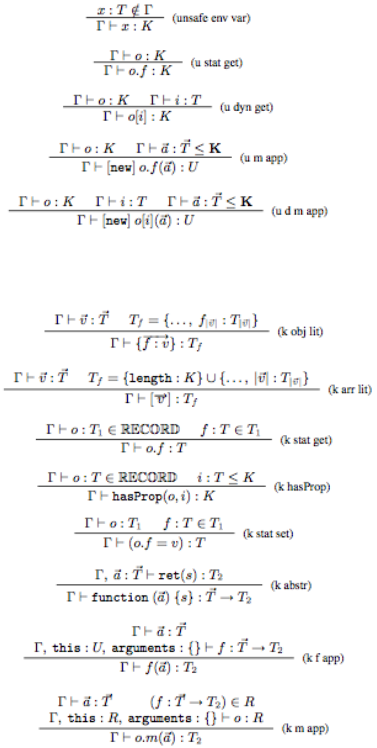
browser support

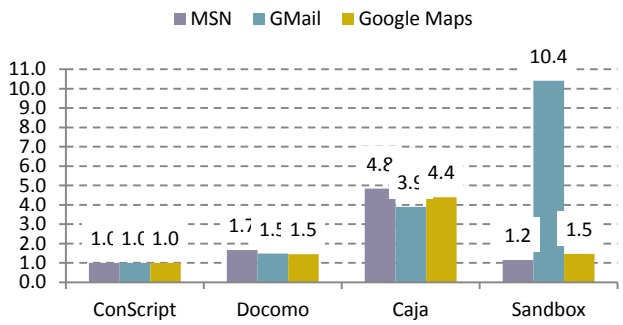
- built into IE 8 JavaScript interpreter
- runtime and space overheads under 1% (vs. 30-550%)
- smaller trusted computing base (TCB)

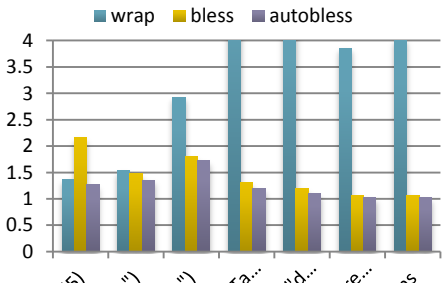
Questions?

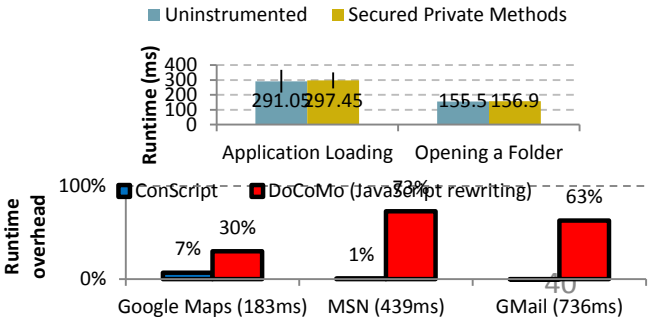

manifest of URLs
limit eval
no foreign links
resource blacklists
no hidden frames
script whitelist
<noscript>
no URL redirection
HTTP-only cookies
no pop-ups
enforce public vs. private






MSN
GMail
Google Maps
11.0
10.0
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
1.0 1.0 1.0
1.7 1.5 1.5
4.8
3.9
4.4
1.2
10.4
1.5
ConScript
Docomo
Caja
Sandbox


wrap
bless
autobless
4
3.5
3
2.5
2
1.5
1
0.5
0


Uninstrumented
Secured Private Methods
Runtime (ms)
400
300
200
100
0
291.05
297.45
155.5
156.9
Application Loading
Opening a Folder
Runtime overhead
100%
0%
ConScript
DoCoMo (Javascript rewriting)
7%
30%
1%
73%
63%
Google Maps (183ms)
MSN (439ms)
GMail (736ms)

END.