ConScript

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

Leo Meyerovich
UC Berkeley

Benjamin Livshits
Microsoft Research
Complications

Benign but buggy: who is to blame?

Downright malicious
Prototype hijacking

Code constantly evolving
How do we maintain quality?
Developer’s Dilemma

Mashups mean including code

Other people’s code can’t be trusted
Only Allow `eval` of JSON

• Idea for a policy:
  – Parse input strings instead of running them
  – Use ConScript to *advise* `eval` calls

• AspectJ advice for Java

```java
void around call Window::eval (String s) { ... }
```

• How to do advice in JavaScript?
  – No classes to speak of
Only Allow eval of JSON

- eval("(xhr.open('evil.com'););")
- eval("([{'hello': 'Oakland'}, 2010]);")
Advising Calls is Tricky

\[
\text{window.eval} = \text{function allowJSON() \{ ... \}}
\]
Advising Calls is Tricky

```
window.eval = function allowJSON() { ... }
```
Advising Calls is Tricky

window.eval = function allowJSON() { ... }
ConScript approach

- Deep advice for complete mediation
- Implemented within the browser for efficiency and reliability
Example of Applying Advice in ConScript

1. `<SCRIPT SRC="facebook.js" POLICY="`
2. `var substr = String.prototype.substring;`
3. `var parse = JSON.parse;`
4. `around(window.eval,`
5. `
   function(oldEval, str) {
5. `    var str2 = uCall(str, substr, 1,
6. `                      str.length - 1);
7. `    var res = parse(str2);
8. `    if (res) return res;
9. `    else throw "eval only for JSON";
10. `   } });"`
# Contributions of ConScript

| A case for aspects in browser | - Protect benign users by giving control to hosting site  
|                              | - ConScript approach: browser-supported aspects |
| Correctness checking         | - Policies are easy to get wrong  
|                              | - Type system to ensure policy correctness |
| Expressiveness               | - Wide catalog of policies from literature and practice  
|                              | - 17 concise hand-written policies  
|                              | - Implemented 2 policy generators |
| Real-world Evaluation        | - Built into IE8 JavaScript interpreter  
|                              | - Tested on real apps: Google Maps, Live Desktop, etc.  
<p>|                              | - Runtime and space overheads under 1% (vs. 30-550%) |</p>
<table>
<thead>
<tr>
<th>A case for aspects in browser</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness checking</td>
<td></td>
</tr>
<tr>
<td>Expressiveness</td>
<td></td>
</tr>
<tr>
<td>Real-world Evaluation</td>
<td></td>
</tr>
</tbody>
</table>
Advising JavaScript Functions in IE8

around(paint, withBoundChecks);
dog.draw();
fish.display();
This is Just the Beginning...

• Not just JavaScript functions
  – native JavaScript calls: `Math.round`, ...
  – DOM calls: `document.getElementById`, ...

• Not just functions...
  – script introduction
  – ...

• Optimizations
  – Blessing
  – Auto-blessing
### A case for aspects in browser

#### Correctness checking

#### Expressiveness

#### Real-world Evaluation

<table>
<thead>
<tr>
<th>Type system</th>
<th>Correctness checking</th>
<th>Expressiveness</th>
<th>Real-world Evaluation</th>
</tr>
</thead>
</table>
Policies are Easy to Get Wrong

1. `var okOrigin="http://www.google.com":true`;
2. `around(window.postMessage,`
3. `function (post, msg, target) {
4.   if (!okOrigin[target]) {
5.     throw 'err';
6.   } else {
7.     return post.call(this, msg, target);
8.   }
9. })`;
1. var okOrigin = {'http://www.google.com': true};
2. around = (window.postMessage, function (post, msg, target) {
    if (!okOrigin[target]) {
        throw 'err';
    } else {
        return post.call(this, msg, target);
    }
});
3. function (post) {
    if (!okOrigin[target]) {
        throw 'err';
    } else {
        return post.call(this, msg, target);
    }
}
4. Object.prototype.poisoning!
5. toString redefinition!
6. Function.prototype.poisoning!
How Do We Enforce Policy Correctness?

Application code

- Unperturbed usage of legacy code

- Disallow `arguments.caller` to avoid stack inspection
  
  (disallowed by ES5’s strict mode)

Policy code

- Modify the JavaScript interpreter
  - introduce `uCall`, `hasProp`, and `toPrimitive`
  - disable `eval`

- Propose a type system to enforce correct use of these primitives
  - disable `with`, ...
Policy Type System

- ML-like type system
- Uses security labels to denote privilege levels
- Enforces *access path integrity* and *reference isolation*

\[
\begin{align*}
\Gamma \vdash o : (f : T^L; r)^o \quad \Gamma \vdash v : T^L \\
\Gamma \vdash o.f = v : T^L
\end{align*}
\]  

(k stat set)
Policy Type System

Reference isolation

• $o$ does not leak through poisoning if $f$ is a field

• Enforces path integrity and reference isolation

\[
\Gamma \vdash o : (f : T^L; r)^o \quad \Gamma \vdash v : T^L
\]

\[
\Gamma \vdash o.f = v : T^L
\]  
(k stat set)

Access path integrity for function calls

• $o.f$ remains unpoisoned if $T$ in $v : T$ is not poisoned
A case for aspects in browser

Correctness checking

Expressiveness

Real-world Evaluation

Policies
ConScript Policies

• 17 hand-written policies
  – Diverse: based on literature, bugs, and anti-patterns
  – Short: wrote new HTML tags *with only a few lines of code*

• 2 automatic policy generators
  – Using runtime analysis
  – Using static analysis
Paper presents 17 ConScript Policies
Paper presents

around(document.createElement,
    function (c : K, tag : U) {
        var elt : U = uCall(document, c, tag);
        if (elt.nodeName == "IFRAME") throw 'err';
        else return elt;
    });
Generating Intrusion Detection Policies

ConScript instrumentation

Eval
new Function(“string”)
postMessage
XDomainRequest
xmlHttpRequest

ConScript enforcement

Observed method calls
Enforcing C# Access Modifiers

class File {
    public File () { ... }
    private open () { ... }
    ...
}

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

c#  JavaScript

Script# compiler
Enforcing C# Access Modifiers

class File {
    public File () { ... } 
    private open () { ... }
    ...
}

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

C#

Script# compiler

JavaScript

flickr

jQuery
class File {
    public File () { ... }
    private open () { ... }
    ...
}

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

C#

Script# compiler

policy generator

JavaScript

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

flickr

jQuery

ConScript
Experimental Evaluation

**Low adoption barrier**

- Implemented on top of the IE 8 JavaScript interpreter
- TCB increase: under 1,000 lines added to IE8’s JavaScript engine
- Changed a few language constructs to disallow `arguments.callee`

**Micro-benchmarks**

- Function, DOM call, eval overhead
- 2.5x faster than previously published source-level wrapping
- Advice optimizations make it faster still

**Macro-benchmarks**

- Google Maps, MSN, Gmail, Live Desktop, Google Calendar
- Hundreds of KB of JavaScript code
- Runtime overhead due to ConScript advice: mostly under 1%
- File size increase due to ConScript advice: under 1%
DoCoMo Policy Enforcement Overhead

File Size Increase for Blacklisting Policy

- ConScript
- Docomo
- Caja
- Sandbox

<table>
<thead>
<tr>
<th></th>
<th>MSN</th>
<th>GMail</th>
<th>Google Maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConScript</td>
<td>1.0</td>
<td>1.7</td>
<td>4.8</td>
</tr>
<tr>
<td>Docomo</td>
<td>1.0</td>
<td>1.5</td>
<td>3.9</td>
</tr>
<tr>
<td>Caja</td>
<td>1.5</td>
<td>4.4</td>
<td>10.4</td>
</tr>
<tr>
<td>Sandbox</td>
<td>1.2</td>
<td>1.5</td>
<td>1.0</td>
</tr>
</tbody>
</table>
## Conclusions

<table>
<thead>
<tr>
<th>A case for aspects in browser</th>
<th>To provide reliable enforcement, browser changes are required and can be minimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness checking</td>
<td>Previous attempts illustrate that hand-written policies are buggy. ConScript addresses this with a type system without affecting legacy code</td>
</tr>
<tr>
<td>Expressiveness</td>
<td>Provide a catalog of 17 hand-written policies for other researchers to use and show how policies can be generated by translators like Script#</td>
</tr>
<tr>
<td>Real-world Evaluation</td>
<td>Implementing policy enforcement in the browser and not at the source level has tremendous performance advantages</td>
</tr>
</tbody>
</table>
QUESTIONS?
Mediating DOM Functions

JavaScript interpreter

IE8 libraries (HTML, Networking, ...)

window.postMessage

frame2.postMessage

IE8 libraries (HTML, Networking, ...)

postMessage

0xff34e5
arguments: "hello", "evil.com"

around(window.postMessage, off);

0xff34e5
[not found]
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