CLIENT-SIDE STATIC ANALYSIS

Ben Livshits, Microsoft Research
Overview of Today’s Lecture

- Client-side JavaScript
  - Analysis of JavaScript
  - `eval` and code obfuscation
  - Need for runtime enforcement

- Gatekeeper as illustration

- Browser
  - Plugins
  - Extensions
    - Firefox extension model
    - Chrome extension model
  - Looking forward
Layers of Browser Security

- JavaScript
- Extension
- plugin
- plugin
- plugin
- browser
- OS
App Store: Centralized Software Distribution

developer

Code

submission

app store

Do checking/verification as part of app approval process
Static Analysis

Last time
- Server-side analysis
- Benign but buggy code

Today
- Client-side analysis
- Buggy or potentially malicious code

Analysis soundness really helps
Primary focus: statically enforcing security and reliability policies for JavaScript code

These policies include semantic properties
  - restricting widget capabilities,
  - making sure built-in objects are not modified,
  - preventing code injection attempts,
  - redirect and cross-site scripting detection,
  - preventing global namespace pollution,
  - taint checking,
  - etc.

Soundly enforcing security policies is hard
Gatekeeper

Mostly Static Enforcement of Security & Reliability Policies for JavaScript Code
alert('hi');

program

malicious
don’t want to allow alert box
can we figure this out statically?

Catch me if you can
alert(‘hi’);

document.write("<script>alert(‘hi’);</script>”);

var d = document;
var w = d.write;
w("<script>alert(‘hi’);”);
eval("do"+"cu"+"ment.write("+...

var e = window.eval;
e("do"+"cu"+"ment.write("...");
var e = new Function("eval");
e.call(
    "do"+"cu"+"ment.write("...");

var e = new Function(unescape("%65%76%61%6C"));
e.call("do"+"cu"+"ment.write("...");
Gatekeeper

Static analysis for JavaScript

- General technology we developed for JavaScript
- Can use for performance optimizations, etc.

This paper

- Use to enforce security and reliability policies
- Analyze Web widgets

Focus on whole program analysis. Contrast with:

- JavaScript language subsets (do a little of)
- JavaScript code rewriting (do a little of)
Goal of Gatekeeper:

Reason about JavaScript code statically

```
alert('hi');
```

Gatekeeper
// register your Gadget's namespace
registerNamespace("GadgetGamez");

// define the constructor for your Gadget (this must match the name in the manifest xml)
GadgetGamez.gg2manybugs = function(p_elSource, p_args, p_namespace) {
    // always call initializeBase before anything else!
    GadgetGamez.gg2manybugs.initializeBase(this, arguments);

    // setup private member variables
    var m_this = this;
    var m_el = p_elSource;
    var m_module = p_args.module;

    /*************************************************************************
    **          initialize Method                                           
    *************************************************************************/
    this.initialize = function(p_objScope) {
        // always call the base object's initialize first!
        GadgetGamez.gg2manybugs.getBaseMethod(this, "initialize", "Web.Bindings.Base").call(this, p_objScope);
        var url = "http://www.gadgetgamez.com/live/2manybugs.htm"
        m_iframe = document.createElement("iframe");
        m_iframe.scrolling = "yes";
        m_iframe.frameBorder = "0";
        m_iframe.src = url;
        m_iframe.width = "95%";
        m_iframe.height = "250px";
        p_elSource.appendChild(m_iframe);
    };
    GadgetGamez.gg2manybugs.registerBaseMethod(this, "initialize");

    /*************************************************************************/
    **           dispose Method                                             
    *************************************************************************/
    this.dispose = function(p_blnUnload) {
        // TODO: add your dispose code here
        // null out all member variables
        m_this = null;
        m_el = null;
        m_module = null;
        // always call the base object's dispose last!
        GadgetGamez.gg2manybugs.getBaseMethod(this, "dispose", "Web.Bindings.Base").call(this, p_blnUnload);
    };
    GadgetGamez.gg2manybugs.registerBaseMethod(this, "dispose");

    /*************************************************************************/
    ** Other Methods                                                       
    *************************************************************************/
};
Sample iGoogle Gadget

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs height="325" title="requestShareApp Example">
    <Require feature="views" />
    <Require feature="opensocial-0.9" />
  </ModulePrefs>
  <Content type="html" view="home, canvas">
    <![CDATA[
      <script type="text/javascript">
        function shareApp() {
          var recipient = null;
          var reason = opensocial.newMessage('Install this gadget');
          opensocial.requestShareApp(recipient, reason, function(response) {
            if (response != null && response.hadError()) {
              alert('requestShareApp Error Code: ' + response.getErrorCode());
            } else if (response != null) {
              alert('requestShareApp OK, Data: ' + gadgets.json.stringify(response));
            } else {
              alert('requestShareApp callback has null response');
            }
          });
        }
      </script>
      <div style="text-align: center">
        <img src="http://gadget-doc-examples.googlecode.com/svn/"
            br/br>
        <input type="button" onclick="shareApp()" value="Share me!">
      </div>
    ]]>
  </Content>
</Module>
```
Widgets are everywhere...

We use over 8,500 widgets to evaluate Gatekeeper
Gatekeeper: Deployment Step on Widget Host

Hosting site: control widgets by enforcing policies:

- No alert
- No redirects
- No document.write

Widget:

...  
alert('hi');  
...
Outline

• Statically analyzable subset JavaScript_{SAFE}

• Points-to analysis for JavaScript

• Formulate nine security & reliability policies

• Experiments
Techniques
Start with Entire JavaScript...

```javascript
var e = new Function("eval");
e.call("do"+"cu"+"ment.write("..."));

var e = new Function(unescape("%65%76%61%6C"));
e.call("do"+"cu"+"ment.write("..."));
```
Remove eval & Friends...

EcmaScript 262

- eval
- setTimeout
- setInterval
- Function
- with
- arguments array

-----------------------

= JavaScript\textsubscript{GK}
Remove Unresolved Array Accesses...

EcmaScript 262

JavaScript$_{GK}$

- innerHTML assignments
- non-const array access $a[x+y]$

= JavaScript$_{SAFE}$

```javascript
var z = 'ev' + x + 'al';
var e = document[z];
```

eval is back!
Now, this is Amenable to Analysis!

EcmaScript 262

JavaScript$_GK$

JavaScript$_SAFE$

\[
\begin{align*}
    s \ ::= & \quad \text{// assignments} \\
            & v1 = v2 \\
            & v = \text{bot} \\
            & \text{return } v \\
            & \text{// calls} \\
            & v = \text{new } v0(v1,\ldots,v_n) \\
            & v = v0(v\text{this},v1,\ldots,v_n) \\
            & \text{// heap} \\
            & v1 = v2.f \\
            & v1.f = v2 \\
            & \text{// declarations} \\
            & v = \text{function}(v1,\ldots,v_n)\{s\}
\end{align*}
\]

JavaScript$_GK$ – need basic instrumentation to prevent runtime code introduction.

JavaScript$_SAFE$ – can analyze fully statically without resorting to runtime checks.
How Many Widgets are in the Subsets?

Ultimately, can analyze 65-97% of all widgets
Sound analysis:

ensures that our policy checkers find all violations
Points-to Analysis in Gatekeeper

% Basic rules
PTSTo(v, h) :- ALLOC(v, h).
PTSTo(v, h) :- FUNCDECL(v, h).
PTSTo(v1, h) :- PTSTo(v2, h), ASSIGN(v1, v2).

DIRECTHEAPSTORESTo(h1, f, h2) :- STORE(v1, f, v2), PTSTo(v1, h1), PTSTo(v2, h2).
DIRECTHEAPPOINTSTo(h1, f, h2) :- DIRECTHEAPSTORESTo(h1, f, h2).
PTSTo(v2, h2) :- LOAD(v2, v1, f), PTSTo(v1, h1), HEAPPTSTo(h1, f, h2).
HEAPPTSTo(h1, f, h2) :- DIRECTHEAPPOINTSTo(h1, f, h2).

% Call graph
CALLS(i, m) :- ACTUAL(i, 0, c), PTSTo(c, m).

% Interprocedural assignments
ASSIGN(v1, v2) :- CALLS(i, m), FORMAL(m, z, v1), ACTUAL(i, z, v2), z > 0.
ASSIGN(v2, v1) :- CALLS(i, m), METHODRET(m, v1), CALLRET(i, v2).

% Prototype handling
HEAPPTSTo(h1, f, h2) :- Prototype(h1, h), HEAPPTSTo(h, f, h2).
Datalog Policy for Preventing `document.write`

1. `DocumentWrite(i) :-`
2. `PointsTo("global", h1),`
3. `HeapPointsTo(h1, "document", h2),`
4. `HeapPointsTo(h2, "write", h3),`
5. `Calls(i, h3).`
EXPERIMENTAL EVALUATION
Policies for Widget Security & Reliability

AlertCalls(i) :- PointsTo("global", h), HeapPointsTo(h, "alert", h2), Calls(i, h2).

DocumentWrite(i) :- PointsTo("global", h1), HeapPointsTo(h1, "document", h2), HeapPointsTo(h2, "write", h3), Calls(i, h3).

DocumentWrite(i) :- PointsTo("global", h1), HeapPointsTo(h1, "document", h2), HeapPointsTo(h2, "writeln", h3), Calls(i, h3).

InnerHTML(v) :- Store(v, "innerHTML", _).

BuiltinObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "String", h).
BuiltinObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "Date", h).
BuiltinObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "Array", h).
BuiltinObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "Boolean", h).
BuiltinObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "Math", h).

BuiltinObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "Function", h).
BuiltinObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "Document", h).
BuiltinObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "Window", h).

Reaches(h1, f, h2) :- HeapPointsTo(h1, f, h2).
Reaches(h1, f, h2) :- HeapPointsTo(h1, _, h), Reaches(h, f, h2).

FrozenViolation(v, h1) :- Store(v, _, _), PointsTo(v, h1), BuiltinObject(h1).
FrozenViolation(v, h1) :- Store(v, _, _), PointsTo(v, h1), BuiltinObject(h2), Reaches(h2, f, h1).

LocationObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "location", h).
WindowObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "window", h).

StoreToLocationObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "window", h2), DirectHeapStoreTo(h2, "location", h).
StoreToLocationObject(h) :- PointsTo("global", h1), HeapPointsTo(h1, "document", h2), DirectHeapStoreTo(h2, "location", h).
StoreToLocationObject(h) :- PointsTo("global", h1), DirectHeapStoreTo(h1, "location", h).

StoreInLocationObject(h) :- LocationObject(h1), DirectHeapStoreTo(h1, _, h).

CallLocationMethod(i) :- LocationObject(h), HeapPointsTo(h, "assign", h1), Calls(i, h1).
CallLocationMethod(i) :- LocationObject(h), HeapPointsTo(h, "reload", h1), Calls(i, h1).
CallLocationMethod(i) :- LocationObject(h), HeapPointsTo(h, "replace", h1), Calls(i, h1).

WindowOpenMethodCall(i) :- WindowObject(h1), HeapPointsTo(h1, "open", h2), Calls(i, h2).
Policy Checking Results

- **Warnings**
  - 1,341 warnings found total
  - Span 684 widgets

- **False positives**
  - 113 false positives
  - 2 widgets

- **Manual inspection effort**
  - Took us about 12 hours to check these
False Positives

• Why not more false positives?
  – Most violations are local
  – But this is policy-specific
    – a global taint policy might produce other results

common.js:

```javascript
function MM_preloadImages() {
  var d=m_Doc;
  if(d.images){
    if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,
    a=MM_preloadImages.arguments;
    for(i=0; i<a.length; i++)
      if (a[i].indexOf("#")!=0){
        d.MM_p[j]=new Image;
        d.MM_p[j++].src=a[i];
      }
  }
}
```
Conclusions

Gatekeeper: Static analysis for JavaScript

Technique: points-to analysis

Focus: analyzing widgets

Results:
- 1,341 policy violations
- False positives affect 2 widgets
Question of the day
What is the difference between browser extensions and browser plugins?
Browser Plugins

- Plugins
  - Flash
  - Adobe PDF reader
  - On their way out?

- Come in different flavors
  - ActiveX
  - Firefox extensions
  - Chrome extensions
Plugin Security

- Plugins are often worst offenders when it comes to security
  - True of malware
  - Of use of DEP/ASLR

- Isolation technologies proposed
  - Run plugins in their own processes
  - Low privilege processes if possible

- Sandboxing techniques
  - Native client
  - XAX
PDF goodness in Chrome
Thursday, November 4, 2010 | 12:20 PM

With every Google Chrome release, we hope to bring new features and improvements that will make your life on the web speedier, simpler, and more secure. Today, we’re excited to introduce the integrated PDF viewer to the beta channel.

PDF is a popular file format that’s used for delivering documents on the web (such as the IRS W-4 tax form). To open a PDF document, you’d typically need to install additional software or a browser plug-in in order to view it in a web browser. With the integrated Chrome PDF viewer now available in Chrome’s beta, you can open a PDF document in Chrome without installing additional software. The PDF document will load as quickly and seamlessly as a normal web page in the browser.

Just like we do with web pages viewed in Chrome, we’ve built in an additional layer of security called the “sandbox” around the Chrome PDF viewer to help protect you from malware and security attacks that are targeted at PDF files. For now, the Chrome PDF viewer is available only in the beta channel, but we look forward to adding more polish and features, as well as making it widely available in the stable channel soon.

Posted by John Abd-El-Malek, Software Engineer
Extension Space: an Overview

- Mozilla Firefox
  - Dominates this space with 1,000s of extensions available
  - Millions of downloads
  - Security is not great: rogue extensions, buggy extensions
  - Relies on a community review process to ensure quality

- Google Chrome
  - Extension manifests
  - Runtime enforcement of manifests within the browser
"content_scripts": [
    {
        "all_frames": true,
        "js": ["blocker.js"],
        "matches": ["http://*/*", "https://*/*"],
        "run_at": "document_start"
    },
    {
        "all_frames": true,
        "js": ["scanner.js"],
        "matches": ["http://*/*", "https://*/*"],
        "run_at": "document_idle"
    }
],

311 of 1,137 featured / popular extensions have access to “your data on all websites”.

**Question:** What do extensions really do?
similar to InPrivate Filtering (IE8), but available on other browsers.

PROTECT YOUR PRIVACY:
Ghostery is built and maintained for users that care about their online privacy, and is engineered to detect and block tracking entities. Ghostery registers with a public, global "GhostRank" ecosystem where you can see who's tracking you so you can defend yourself.

James  Jul 19, 2010  Mark as spam
Is this a scam? Shi*, I so nearly downloaded it. Why does it have so many good reviews then?

Jul 8, 2010  Mark as spam
DO NOT DOWNLOAD!!! SCAM & PHISHING EXTENSION!!!

anonymous  Jul 5, 2010  Mark as spam
this is stoopid

asmp  Jul 5, 2010  Mark as spam
This extension is dangerous... blog.betteradvertising.com/2010/01/19/better-advertising-acquires-ghostery/ man! To think I was almost installing this piece of sh.......

GhostRank data is anonymous, it is NEVER used for advertising targeting purposes, and is only shared in an aggregated, non-personal, statistical form.
311 of 1,137 featured / popular extensions have access to “your data on all websites”.

Install JavaScript ToolBox?
This extension needs access to:
Your data on all websites

Install InvisibleHand?
This extension needs access to:
Your data on all websites
Your browsing history

Install Typograf?
This extension needs access to:
Your data on all websites

Install Note Anywhere?
This extension needs access to:
Your data on all websites
Your browsing history

Install Short URL Scanner?
This extension needs access to:
Your data on all websites
Your browsing history

Install Google Mail Checker Plus?
This extension needs access to:
Your data on all websites
Your browsing history
Verified Security for Browser Extensions

Nikhil Swamy

With Arjun Guha, Matthew Fredrikson, and Ben Livshits

[Oakland S&P, 2011]
arbitrary (Apple)
too permissive (Mozilla)

submit

reject

accept

cross-platform extensions are hard
precise policy +

predictable, reliable

submit

reject

make this easier

accept

Developer

1. No runtime security checks (fast)
2. No security exceptions (robust)

Curator

1. Automate policy compliance checking
2. Tools to understand policies

Users

precise policy +
let name = document.getName() in
let website = document.getWebsites()[0] in ...

getName and getWebsites do not exist...
<table>
<thead>
<tr>
<th>Label</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email:</td>
<td><a href="mailto:jean.yang.writeme@gmail.com">jean.yang.writeme@gmail.com</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:jeanyang@mit.edu">jeanyang@mit.edu</a></td>
</tr>
<tr>
<td>Mobile Phone:</td>
<td>[Redacted]</td>
</tr>
<tr>
<td>Website:</td>
<td>[Hyperlinks: people.csail.mit.edu/jeanyang, people.csail.mit.edu/jeanyang, jxyzabc.blogspot.com, gsc.mit.edu/gwamit]</td>
</tr>
</tbody>
</table>
Policy: can read `<td class="data">` tags, which have a sibling `<td class="label">Website:`</td>
Summary

- Client-side JavaScript
  - Analysis of JavaScript
  - `eval` and code obfuscation
  - Need for runtime enforcement

- Gatekeeper as illustration

- Browser
  - Plugins

- Extensions
  - Firefox extension model
  - Chrome extension model

- Looking forward