Secure Applications Using Hybrid Analysis

Jitendra Sharma
Application Security Specialist
IBM Rational Application Security
sharma.jitendra@in.ibm.com

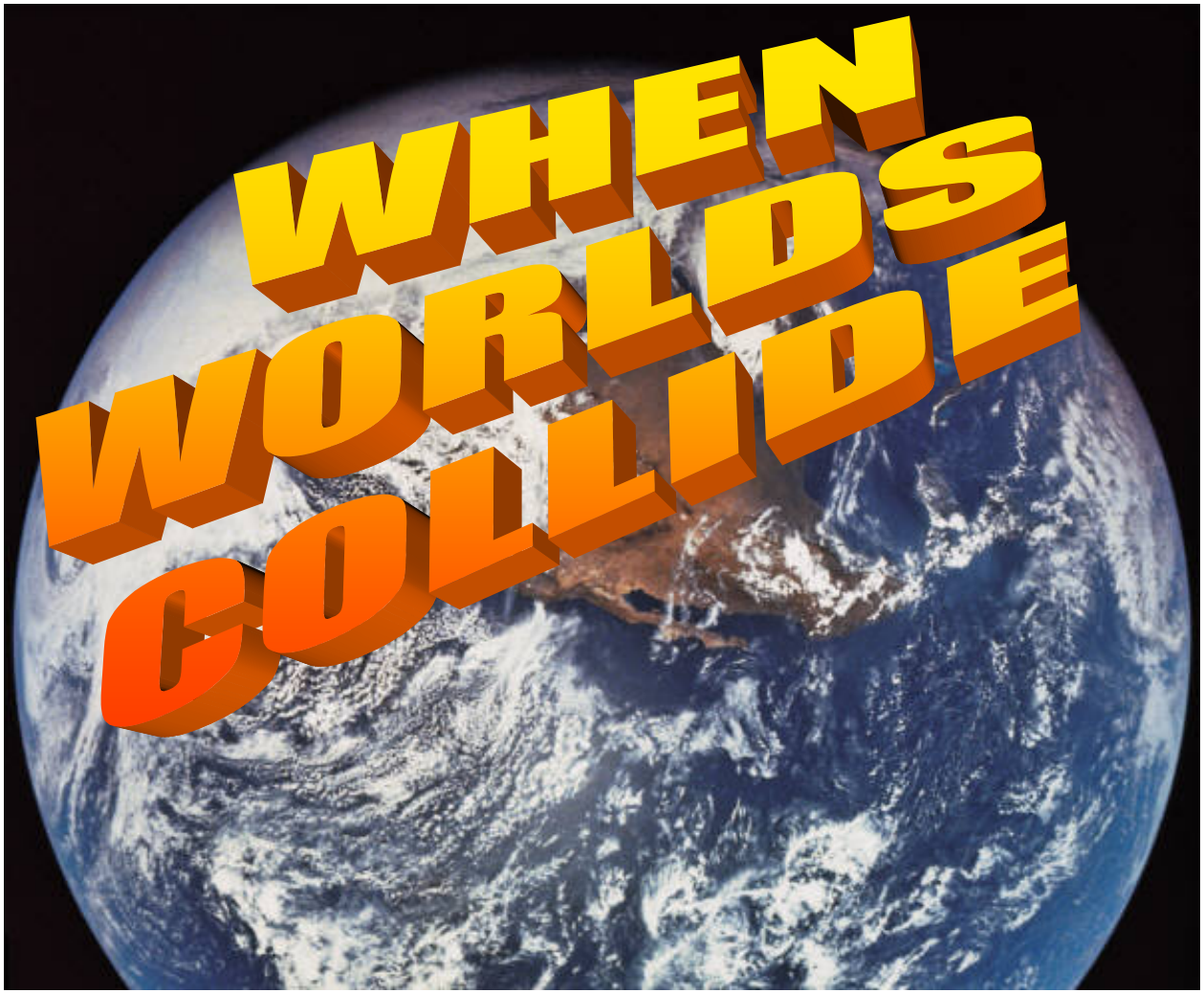
IBM Software
Innovate2011
The Premier Event for Software and Systems Innovation
Software. Everyware.
August 9-11, Bangalore | **August 11,** Delhi

Please Note:

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Hybrid Analysis



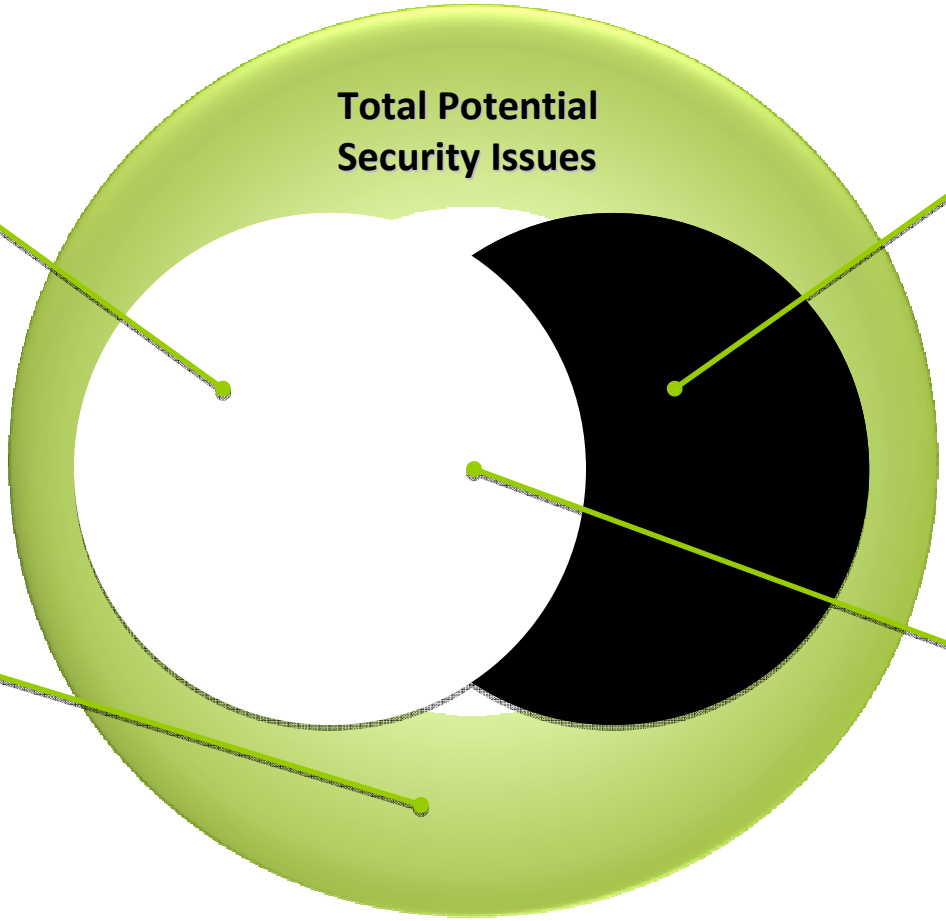
How is this web app vulnerable?

How is this code vulnerable?

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOExceptio
    String step = (request.getParameter("step"));
    if (step == null)
        step = "";

    String content = null;
    if (step.equals("a")){
        content = "<h1>Question 1</h1>"+
        "<div width=\"99%\"><p>Which of the following groups includes your age?<ul>  <li><a href=\"survey_question
    }
    else if (step.equals("done")){
        content = "<h1>Thanks</h1>"+
        "<div width=\"99%\"><p>We will contact you shortly at:<br /><br /> <b>" + request.getParameter("txtEmail")
    }
    else {
        content = "<h1>Welcome</h1>"+
        "<div width=\"99%\"><p>If you complete this survey, you have an opportunity to win an iPod.  Would you lik
    }
    response.setContentType("text/html");
    response.getWriter().write(content);
    response.getWriter().flush();
}
```

Dynamic vs. Static Analysis



Static Analysis

- Null pointer dereference
- Threading issues
- Code quality issues
- Issues in dead code
- Insecure crypto functions
- Issues in back-end application code
- Stored vulneabilities
- Complex SQL injection

Dynamic Analysis

- Environment configuration issues
- Patch level issues
- Runtime privileges issues
- Authentication issues
- Protocol parser/serializer issues
- Session management issues
- Issues in external 3rd party web components
- Cross-site request forgery
- Malware analysis

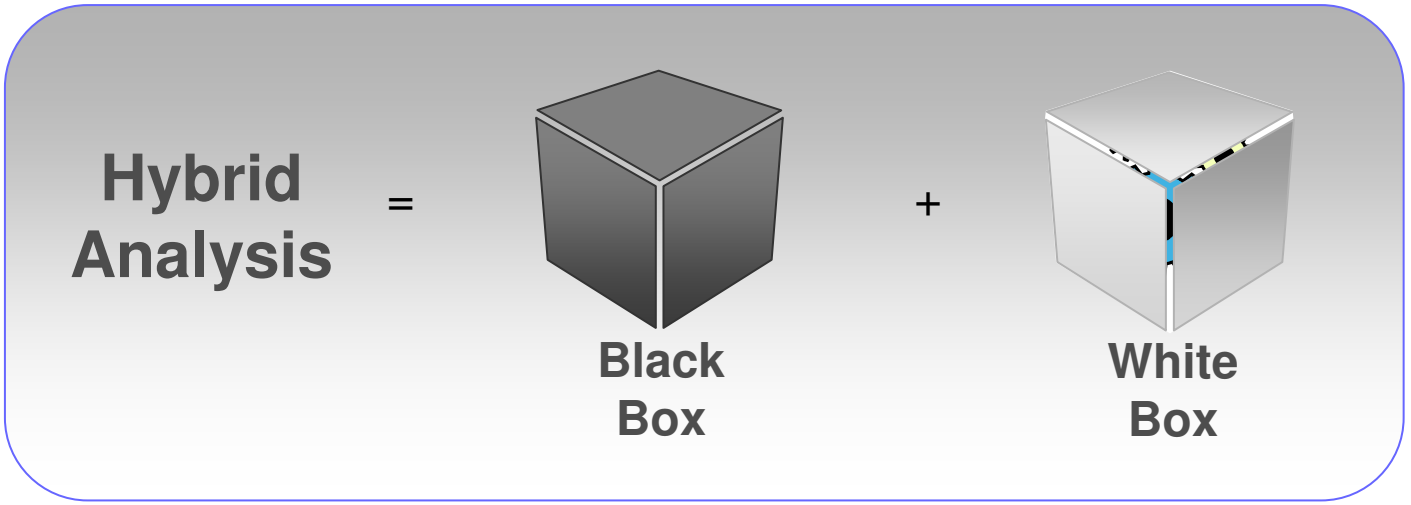
Manual Assessment

- Business logic issues

Dynamic & Static

- SQL Injection
- Cross Site Scripting
- HTTP Response Splitting
- OS Commanding
- LDAP Injection
- XPath Injection
- Path Traversal
- Buffer Overflows
- Format String Issues

Hybrid Analysis



Hybrid Analysis Technologies in IBM Rational AppScan:

- Correlation
- JSA (JavaScript Security Analyzer)

Correlation & Aggregation

Dynamic Analysis issues
AppScan Standard
AppScan Enterprise

+

Static Analysis issues
AppScan Source

Correlated and/or Aggregated issues
AppScan Enterprise
AppScan Reporting Console

		Test URL	Element	Issue Type	Source File	API	Line
☐		http://duncans-xpd:8080/altoromutual/doLogin	uid	Blind SQL Injection	%AltoroJ%\src\main\java\com\ibm\rational\appscan\altoromutual\util\DBUtil.java	java.sql.Statement.executeQuery	112
☐		http://duncans-xpd:8080/altoromutual/doLogin	passw	Blind SQL Injection	%AltoroJ%\src\main\java\com\ibm\rational\appscan\altoromutual\util\DBUtil.java	java.sql.Statement.executeQuery	112
☐		http://duncans-xpd:8080/altoromutual/doLogin	uid	Blind SQL Injection	%AltoroJ%\src\main\java\com\ibm\rational		
☐		http://duncans-xpd:8080/altoromutual/bank/customize.jsp	lang	Cross-Site Scripting	%AltoroJ%\target\AltoroJ_mvn\bank\custo		
☐		http://duncans-xpd:8080/altoromutual/bank/queryxpath.jsp	query	Cross-Site Scripting	%AltoroJ%\target\AltoroJ_mvn\bank\query		
☐		http://duncans-xpd:8080/altoromutual/search.jsp	query	Cross-Site Scripting	%AltoroJ%\target\AltoroJ_mvn\search.jsp		
☐		http://duncans-xpd:8080/altoromutual/admin/addAccount	username	Database Error Pattern Found	%AltoroJ%\src\main\java\com\ibm\rational		
☐		http://duncans-xpd:8080/altoromutual/admin/addAccount	accttypes	Database Error Pattern Found	%AltoroJ%\src\main\java\com\ibm\rational		
☐		http://duncans-xpd:8080/altoromutual/admin/addAccount	username	Database Error Pattern Found	%AltoroJ%\src\main\java\com\ibm\rational		
☐		http://duncans-xpd:8080/altoromutual/admin/addAccount	username	Database Error Pattern Found	%AltoroJ%\src\main\java\com\ibm\rational		
☐		http://duncans-xpd:8080/altoromutual/bank/showAccount	listAccounts	Link Injection (facilitates Cross-Site Request For	%AltoroJ%\src\main\java\com\ibm\rational		
☐		http://duncans-xpd:8080/altoromutual/search.jsp	query	Link Injection (facilitates Cross-Site Request For	%AltoroJ%\target\AltoroJ_mvn\search.jsp		
☐		http://duncans-xpd:8080/altoromutual/bank/queryxpath.jsp	query	Link Injection (facilitates Cross-Site Request For	%AltoroJ%\target\AltoroJ_mvn\bank\query		
☐		http://duncans-xpd:8080/altoromutual/bank/customize.jsp	lang	Link Injection (facilitates Cross-Site Request For	%AltoroJ%\target\AltoroJ_mvn\bank\custo		
☐		http://duncans-xpd:8080/altoromutual/admin/addAccount	username	SQL Injection	%AltoroJ%\src\main\java\com\ibm\rational		
☐		http://duncans-xpd:8080/altoromutual/admin/addAccount	accttypes	SQL Injection	%AltoroJ%\src\main\java\com\ibm\rational\appscan\altoromutual\util\DBUtil.java	java.sql.Statement.execute	327
☐		http://duncans-xpd:8080/altoromutual/admin/addAccount	username	SQL Injection	%AltoroJ%\src\main\java\com\ibm\rational\appscan\altoromutual\util\DBUtil.java	java.sql.Statement.execute	350
☐		http://duncans-xpd:8080/altoromutual/admin/addAccount	username	SQL Injection	%AltoroJ%\src\main\java\com\ibm\rational\appscan\altoromutual\util\DBUtil.java	java.sql.Statement.execute	338

- ✓ Higher confidence
- ✓ Fewer issues to triage
- ✓ All issues in a single location
- ✓ Easier to fix
  (source code location + reproduction scenario)

Aggregated Issues in Compliance Reports

- Compliance reports in AppScan Reporting Console and AppScan Enterprise include both dynamic and static analysis issues to give a complete view of an organization's security compliance

The Payment Card Industry Data Security Standard (PCI)

Export Email

Last Updated: 5/6/2011 10:20:20 AM

Summary | Group | Show | Search | Layout

There are **22** issues making you non-compliant with 40 regulation(s)

All items

Items 1-22 of 22

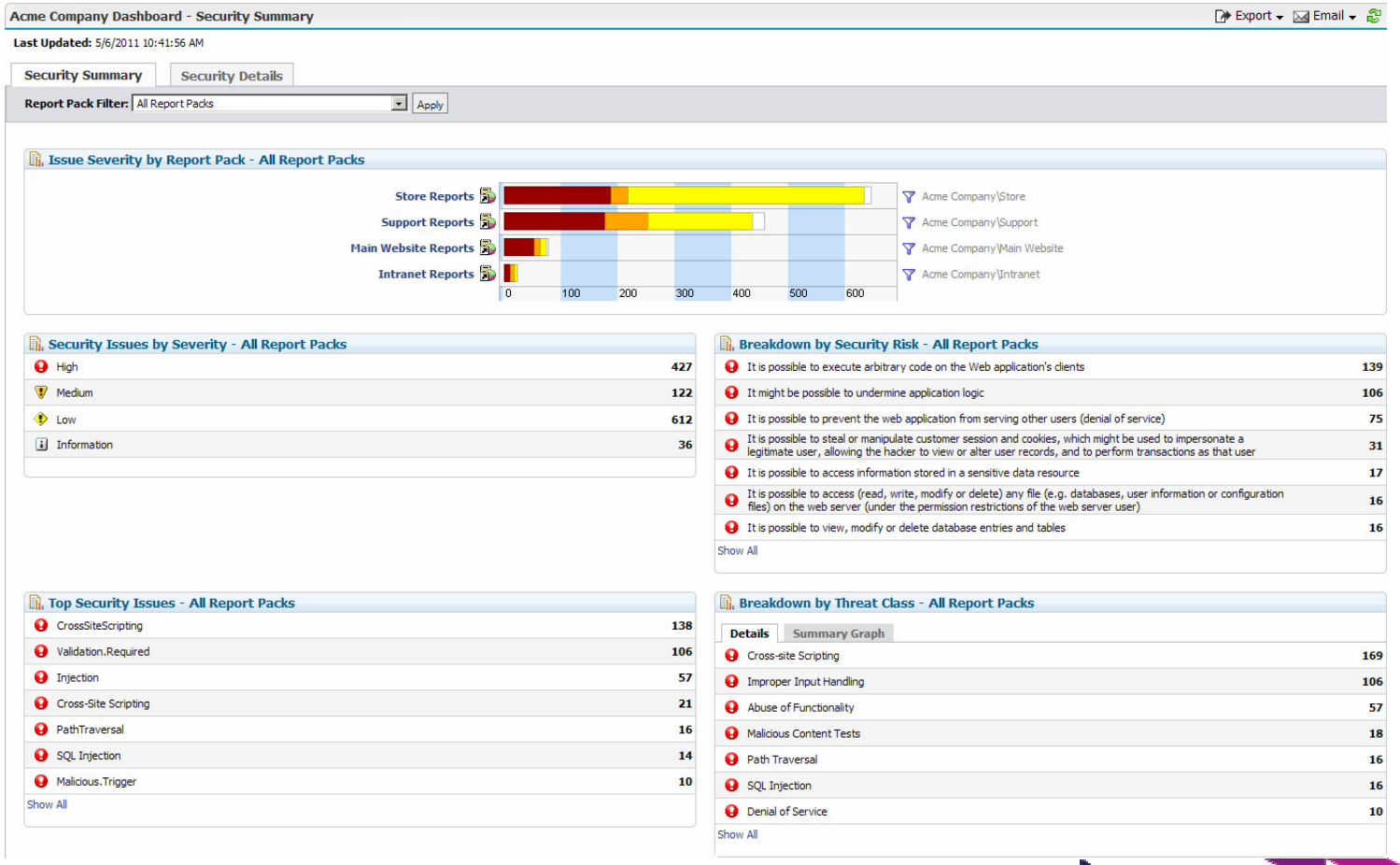
Go to page: 1 of 1 Apply

Action: Export to Excel Apply

Issue	Issue Type	Test URL	Element	Source File	API	Regulation
5939*	Poison Null Byte Windows Files Retrieval	http://revelation/acmehackme/bank/content.aspx	content			Requirement 2.4, Requirement A.1.3, R
5962*	Microsoft IIS .printer Buffer Overflow	http://revelation/acmehackme/NULL.printer				Requirement 3, Requirement 6, Require
144*	PathTraversal			%testapps%\windows\csharp\sharpPDF\Fonts\TTF\IO\Advanc	System.IO.FileStream.FileStream	Requirement 2.4, Requirement A.1.1, R
6065*	Cross-Site Scripting	http://revelation/acmehackme/bank/login.aspx	uid			Requirement 2.4, Requirement A.1.3, R
6091*	Cross-Site Scripting	http://revelation/acmehackme/bank/search.aspx	searchterms			Requirement 2.4, Requirement A.1.3, R
6040*	Unencrypted Login Request	http://revelation/acmehackme/bank/login.aspx	passw			Requirement 2.3, Requirement 2.4, Rec
6082*	Predictable Login Credentials	http://revelation/acmehackme/bank/login.aspx				Requirement 2, Requirement 2.1, Requi
6084*	IIS localstart.asp Possible Brute Force	http://revelation/localstart.asp				Requirement 2, Requirement 2.2.4, Rec
5850*	SQL Injection	http://revelation/acmehackme/bank/login.aspx	uid			Requirement 2.4, Requirement A.1.3, R
5846*	Authentication Bypass Using SQL Injection	http://revelation/acmehackme/bank/login.aspx	passw			Requirement 2.4, Requirement A.1.3, R
6039*	Cross-Site Request Forgery	http://revelation/acmehackme/bank/login.aspx				Requirement 2.4, Requirement A.1.3, R
6064*	Session Identifier Not Updated	http://revelation/acmehackme/bank/login.aspx				Requirement 2, Requirement 2.1, Requi
5989*	Inadequate Account Lockout	http://revelation/acmehackme/bank/login.aspx	passw			Requirement 2, Requirement 2.1, Requi
141*	Validation.Required			%testapps%\windows\csharp\sharpPDF\pdfDocument.cs	System.IO.Stream.Write	Requirement 6, Requirement 6.3, Requi
49*	Injection			%testapps%\windows\csharp\sharpPDF\pdfDocument.cs	System.IO.Stream.Write	Requirement 6, Requirement 6.3, Requi
177*	ErrorHandling.RevealDetails.Message			%testapps%\windows\csharp\sharpPDF\Fonts\AFM\afmFontRe	System.Exception.get_Message	Requirement 2.4, Requirement A.1.1, R
5954*	Directory Listing	http://revelation/acmehackme/admin/				Requirement 2.2.2, Requirement 2.2.3,
5955*	Hidden Directory Detected	http://revelation/_private/				Requirement 2, Requirement 2.1, Requi
111*	Logging.Required			%testapps%\windows\csharp\sharpPDF\Fonts\TTF\IO\Advanc	System.Exception.get_Message	Requirement 2, Requirement 2.1, Requi
5981*	Direct Access to Administration Pages	http://revelation/acmehackme/admin/admin.aspx				Requirement 2, Requirement 2.1, Requi
5998*	Application Input Restrictions Bypass	http://revelation/acmehackme/bank/account.aspx	listAccounts			Requirement 2.4, Requirement A.1.1, R
5883*	Database Error Pattern Found	http://revelation/acmehackme/bank/login.aspx	passw			Requirement 2.4, Requirement A.1.3, R

Aggregated Issues in Dashboards

- Dashboards in AppScan Reporting Console and AppScan Enterprise include both dynamic and static analysis issues to allow management to get an overall picture of their organization's security health



Attack Vector Detection

IBM T.J.Watson Research Lab

- Maps static analysis issues to their associated URL and parameter name

- Correlation
- Cross-validation
- Uniform presentation

WB view: XSS at customize.java line 12

↓ Attack Vector Detection does the mapping

BB view: XSS at account.jsp, parameter “lang”

The same issue can look differently from a WB or BB point of view

Hybrid Analysis for Client-Side Security Motivation

Three Types of XSS

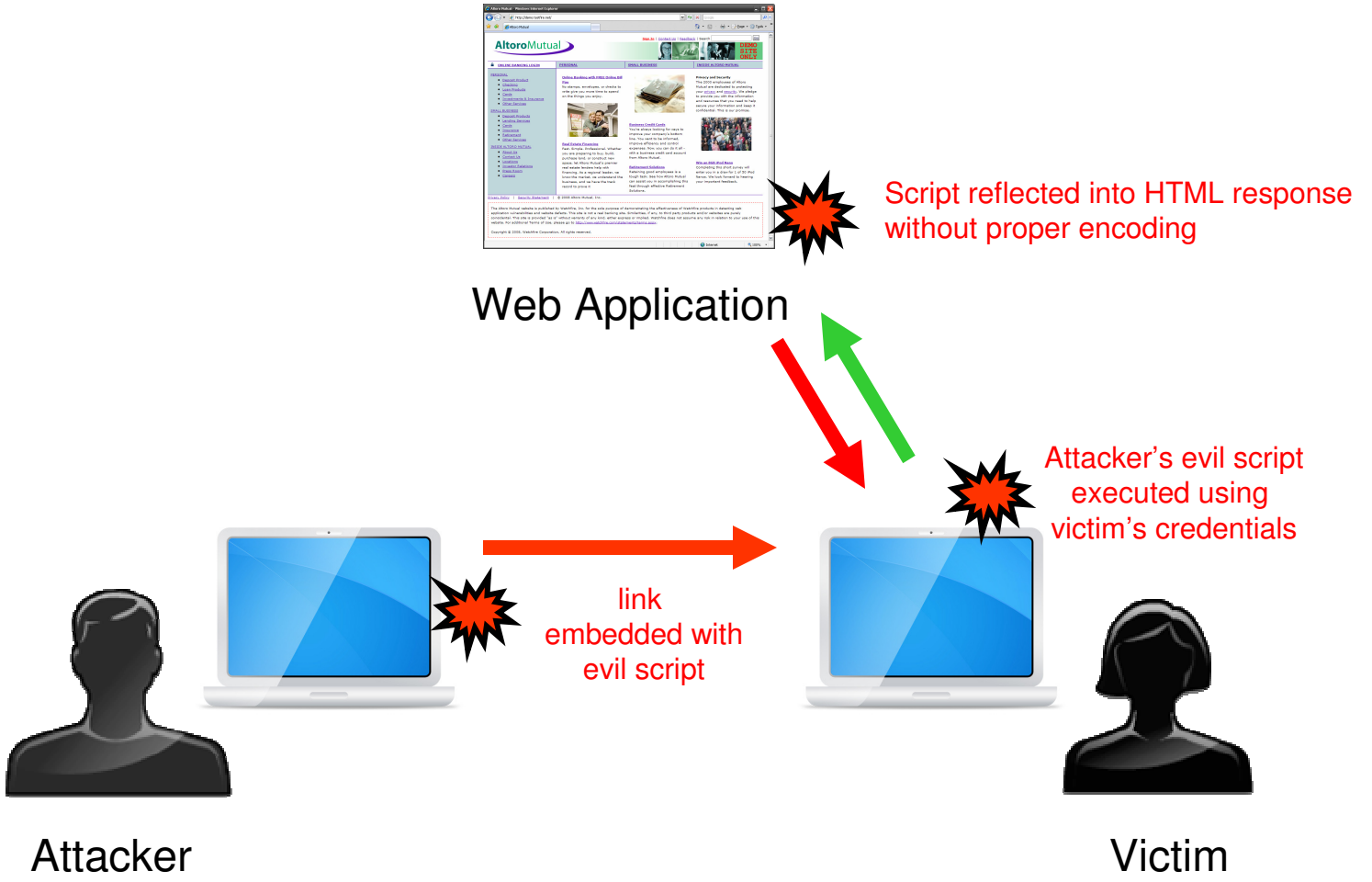
1. *Reflected*
2. *Stored*
3. *DOM-based* **TODAY'S FOCUS**

http://demo.testfire.net/disclaimer.htm?url=http://www.netscape.com

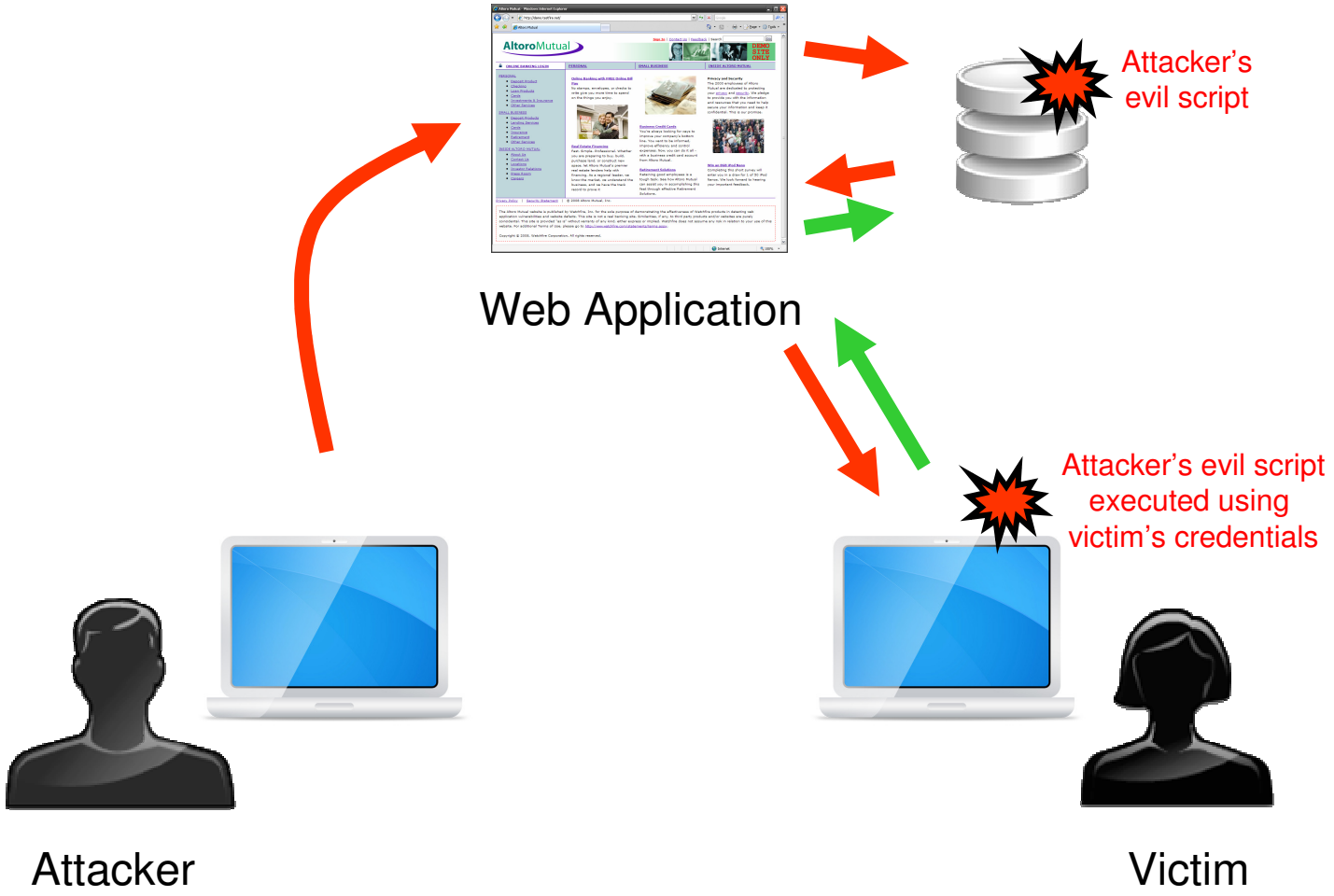
```
19          }
20
21          var iPos = document.URL.indexOf("url=")+4;
22  1       var sDst = document.URL.substring(iPos,document.URL.length);
23        </script>
24      </head>
        ...
31            <td>
32              <p>This hyperlink allows you to access a third party website:
33              <br /><br />
34  2           <b><script>document.write(unescape(sDst));</script></b>
35              <br /><br />
36              Please read the privacy policy of the linked website, which
37              may differ from the privacy policy of the Altoro Mutual website.
```

Example of trace provided by JSA

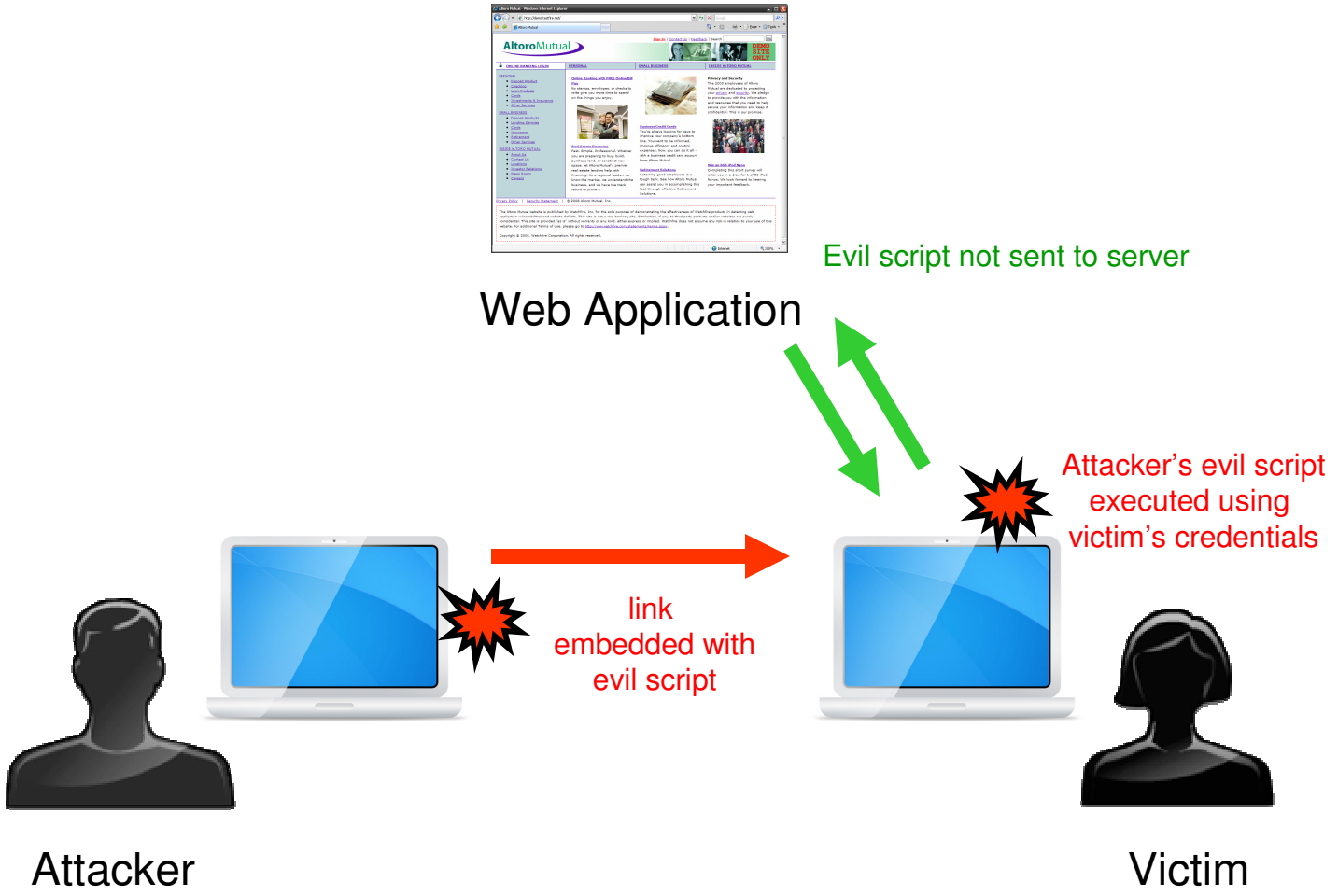
Reflected XSS



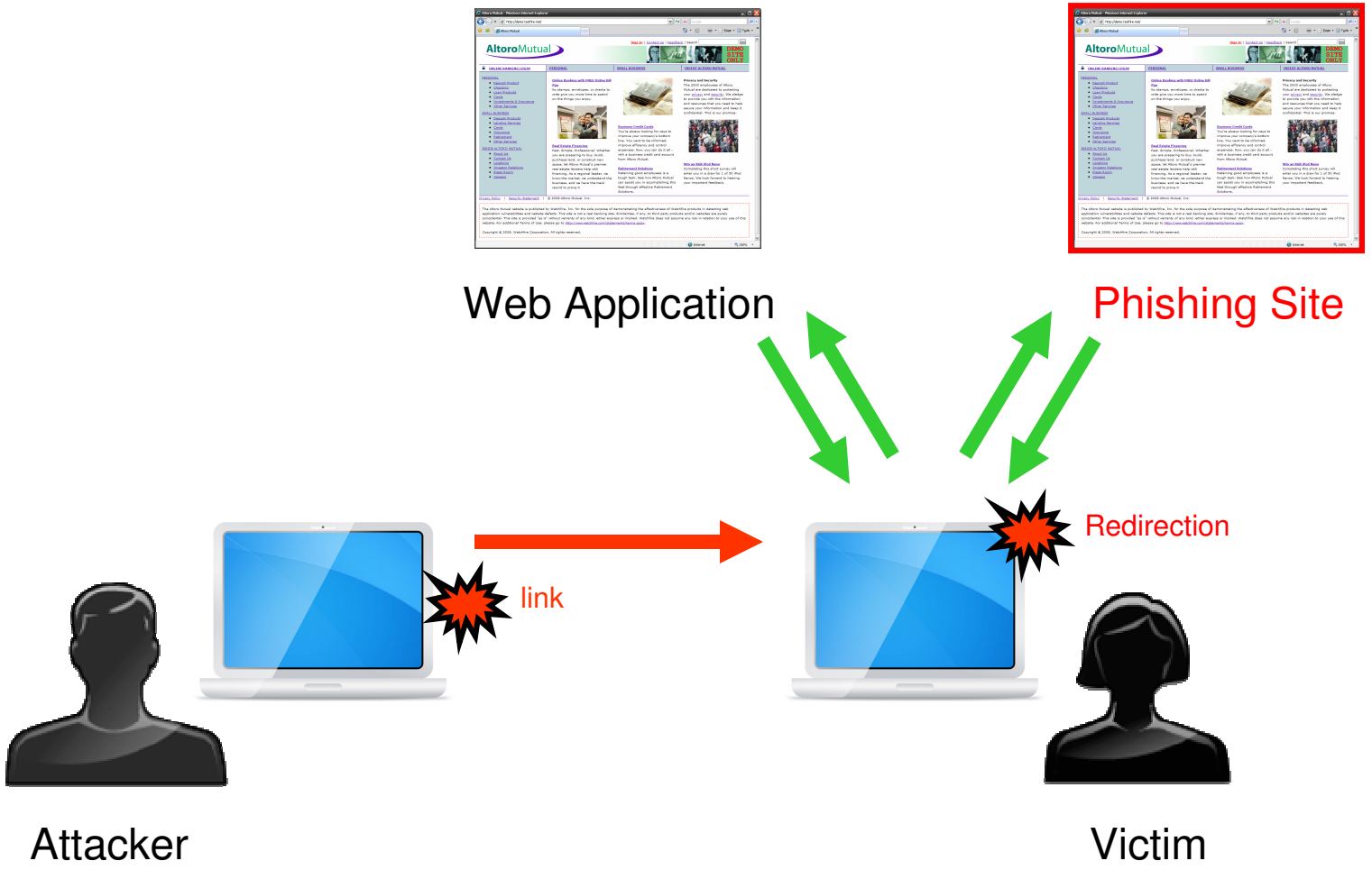
Stored XSS



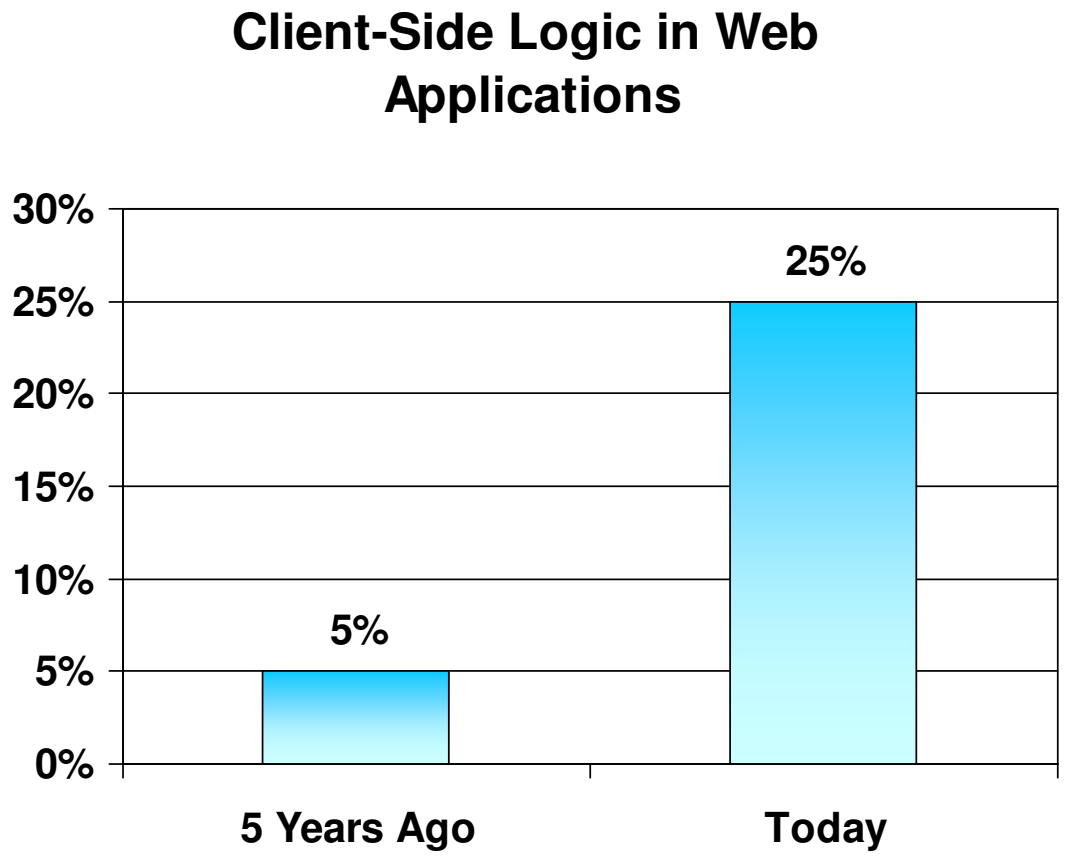
DOM-Based XSS



Open Redirect



Logic Moving to the Client-Side



→ **Challenge for black-box, white-box testing**

Security Problems in JavaScript

AMERICA'S LARGEST CORPORATIONS

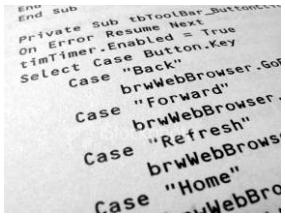
FORTUNE 500

15%

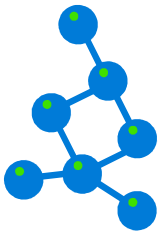
of Fortune 500 websites have exploitable security issues in JavaScript.

According to an IBM study performed in 2010

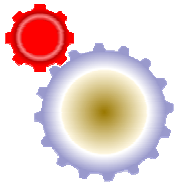
Static Analysis



Source Code



Model

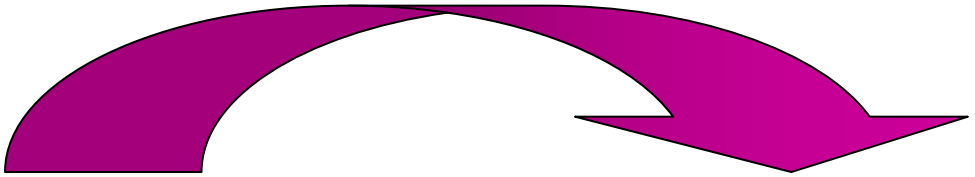


Perform Analysis

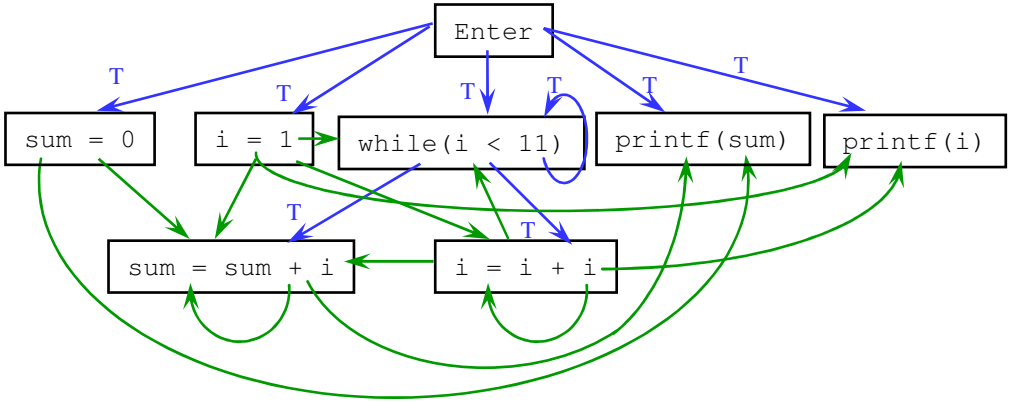
Results

Domain Knowledge (e.g. Security Rules)

Modeling

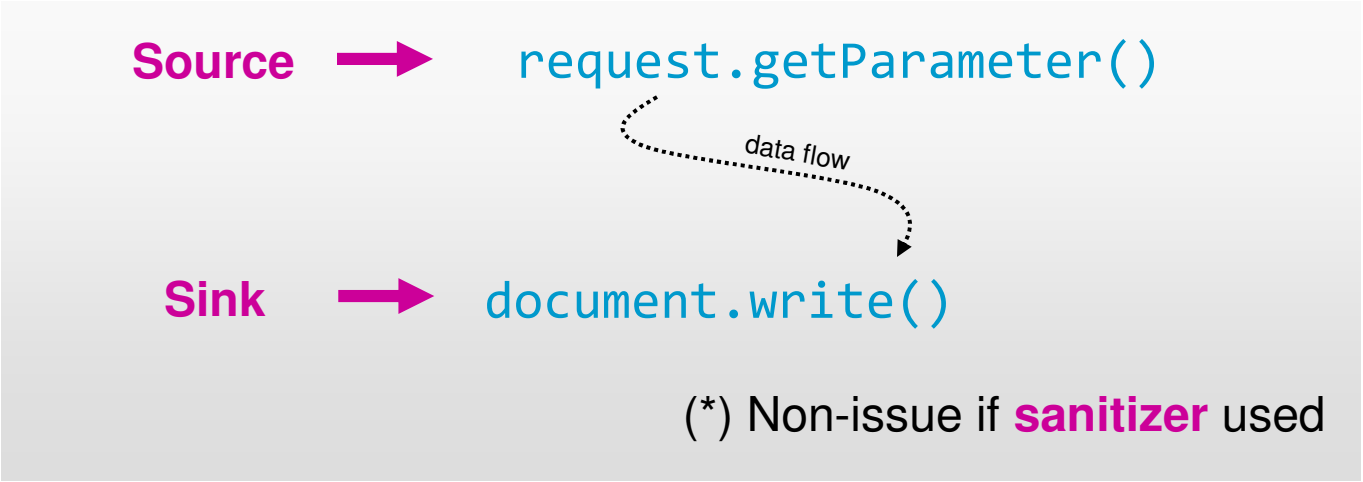


```
int main() {
    int sum = 0;
    int i = 1;
    while (i < 11) {
        sum = sum + i;
        i = i + 1;
    }
    printf("%d\n",sum);
    printf("%d\n",i);
}
```



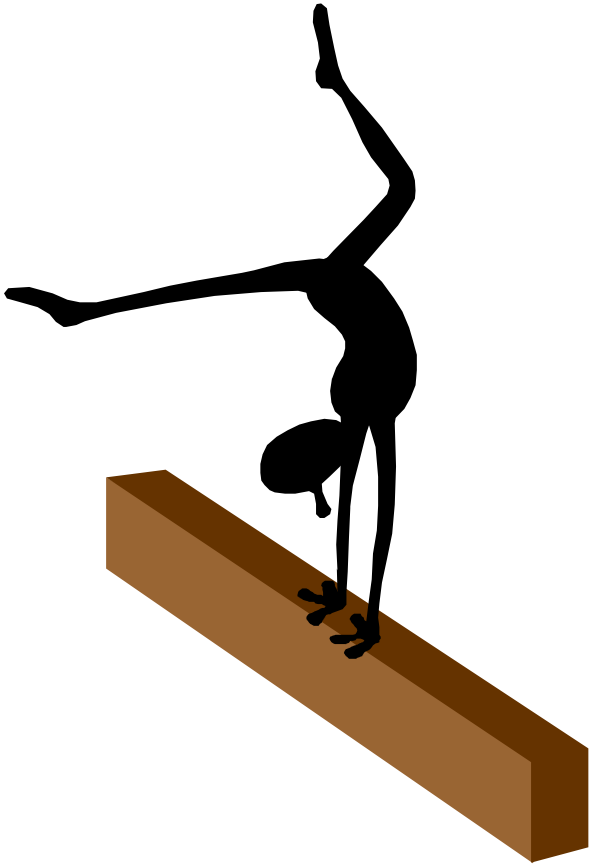
Taint Analysis

- Information-flow violation problems can be solved using **static taint analysis**



- Limitation: binary analysis – data is either tainted or not

Challenges in Static Analysis (1)

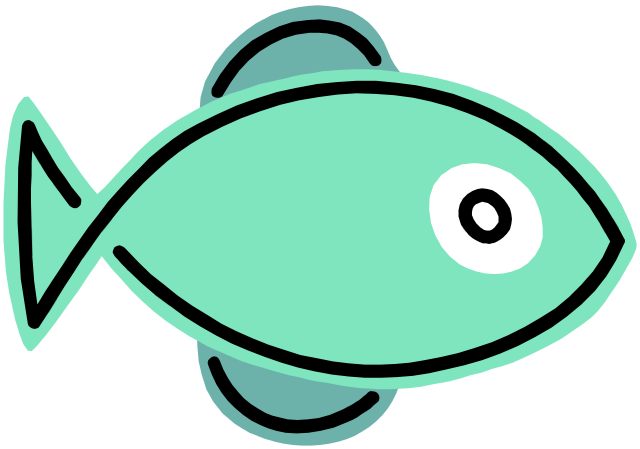


The Balancing Act

Tradeoffs

- Large models or small?
- Faster analysis or more accurate?
- Bias towards false positives or false negatives?

Challenges in Static Analysis (2)



The Babel Fish

Abstraction

Speaking the right language, picking the right abstraction.

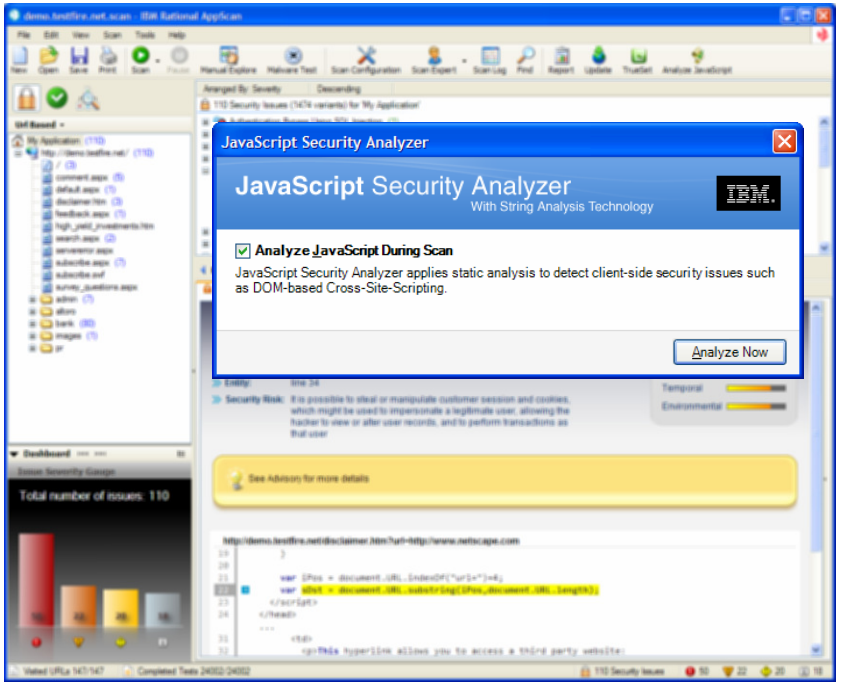
- Taint analysis is a binary analysis: either tainted or not
- But accurate security assessment requires understanding of **string** content and context

Introducing **JavaScript Security Analyzer** (JSA)

What is JSA?

- JavaScript Security Analyzer – An extension of AppScan Standard, developed in collaboration with IBM Research, that does **static taint analysis** of **JavaScript**, detecting a range of client-side security issues:

•DOM Based Cross-Site Scripting
•Web Worker Script URL Manipulation
•Phishing Through URL Redirection
•Notification Phishing
•Email Attribute Spoofing
•Client-Side Stored Cross-Site Scripting

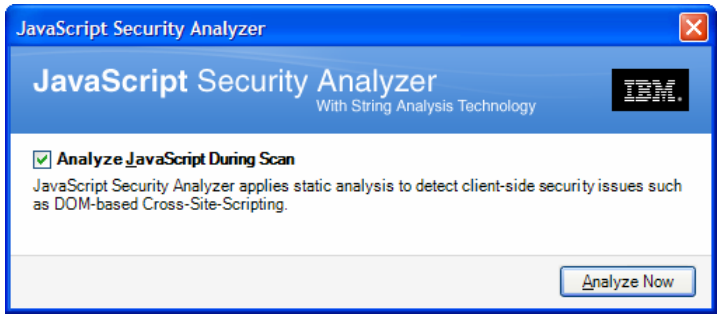


Why is this significant?

- The role of JavaScript in modern web applications becomes greater as technologies such as AJAX, Dojo and HTML5 become more prolific.
- It makes AppScan the **first tool in the world** capable of detecting a range of client-side security issues. These issues are very common but no other tool exists today that can find them.
- JSA makes AppScan the **first scanner that applies BB and WB in the same scan**.
- JSA completes a missing piece in scanning modern web applications. JSX provides an answer for crawling, JSA provides an answer for testing. In the future we see great potential for **synergy between JSX and JSA**.

TAINT analysis | STRING /*analysis*/ | DE-OBFUSCATION | WALA T. J. WATSON LIBRARIES FOR ANALYSIS
Java POWERED | AXF AppScan eXtension Framework | IBM Research | Rational software

How To Run JSA



- **Run a normal scan with AppScan Standard.**
- **JSA works behind the scenes, analyzing JS content in all visited URLs.**
  - No configuration required.
  - Any issues found are added to the issue list, just like other AppScan issues.
  - Issues appear in the GUI, the reports etc.
  - Data-flow trace is provided for the issues found.
  - JSA activity appears in the scan log.
  - JSA runs in parallel to the test stage, in a separate process.

http://demo.testfire.net/disclaimer.htm?url=http://www.netscape.com

```
19          }
20
21          var iPos = document.URL.indexOf("url=")+4;
22  1       var sDst = document.URL.substring(iPos,document.URL.length);
23      </script>
24  </head>
    ...
31          <td>
32            <p>This hyperlink allows you to access a third party website:
33            <br /><br />
34  2         <b><script>document.write(unescape(sDst));</script></b>
35            <br /><br />
36            Please read the privacy policy of the linked website, which
37            may differ from the privacy policy of the Altoro Mutual website.
```

Example of trace provided by JSA

- **JSA can run on demand (“Analyze Now”) or automatically as part of scan (every time test stage starts).**
  - JSA is smart and will not analyze the same content more than once, even if AppScan visited the same page several times.
- ***TIP: you can also apply JSA to your existing scan files!***

Notable Features in JSA

- **HTML5 support**
  - World's first and only tool to analyze and detect client-side security issues in HTML5
- **String Analysis**
  - Enables the engine to eliminate many non-exploitable issues automatically, and to detect other issues more accurately
- **De-obfuscation**
  - De-obfuscation is now integrated into the engine. When JSA finds issues in JS files that are obfuscated or packed, the code is automatically de-obfuscated before the results are presented, making issues easier to understand and triage

String Analysis in JSA

- Used to verify the exploitability of issues found by taint analysis
- Solves almost all false positives (non-exploitable results) in JSA
- Is sound; never eliminates true positives

Motivating Example

```
39          if (url=="popups/emta.asp" || url=="../popups/emta.asp")
40          {
41  (1)         url= url + "?l=" + window.location
42          }                                          SOURCE
43             lft = (screen.availWidth -w)/2;
44             t = (screen.availHeight -h)/2;
45             p = "scrollbars=" + sb + ",resizable=" + rs + ",status=" + st
46  (2)     window.open(url, rand(101)-1, p);
47  SINK    }
48          function rand(number) {
49              return Math.ceil(rnd()*number);
```

Real world example of JavaScript issue detected by taint analysis

- Taint analysis is not “smart” enough. Sometimes, even when there is taint flow, issues are **not exploitable**.
- In this case, tainted data flows into a ‘**window.open**’ command, supposedly allowing an attacker to redirect to a malicious site (**phishing**).
- However, the tainted string is appended AFTER the original URL and a ‘?’ character. Therefore, the **target hostname** is not controlled by the attacker.
- The issue is not exploitable. How do we detect that automatically?

String Analysis in JSA

- **String Analysis in JSA** is used for eliminating false findings. It models strings as a concrete prefix and an unknown suffix, which is a natural fit for taint analysis: The part controlled by the attacker is unknown, but the uncontrolled prefix is modeled precisely.

URL as Source

`http://www.mysite.com/folder/page?a=1&b=2#anchor`

NOT CONTROLLED BY ATTACKER | CONTROLLED BY ATTACKER

- If String Analysis determines that at the point of the sink, the host and path parts are both fixed and not controlled by an attacker, the issue is eliminated. **This is a very powerful analysis**, allowing **superior accuracy** when it comes to ruling out non-exploitable issues:

```
function leaving(){

      var result, search_term = 'login.html';
      var replaceStr = 'login.jsp';
      var str = document.URL;
      var url_check = str.indexOf( search_term );
      if (url_check > -1) {
          result = str.substring(0,url_check);
          result = result + replaceStr +
                   str.substring((url_check+search_term.length),str.length);
          document.URL = result; }}
```

More Uses for String Analysis in JSA

http://www.morganstanley.com/js/ms.js

```
221         var subject = "Information from morganstanley.com";
222         var message = "Please read the article below from morganstanley.com:";
223         var newline = escape("\n\n");
224 (1)     var link = document.location.href;
225
226 (2)     var mymsg = "mailto:" + email + "?subject=" + subject + "&body=" + message + newline + link;
227 (3)     document.location.href = mymsg;
228         //alert (mymsg);
```

- In this real-world example, **String Analysis** can tell us that the target URL always begins with "**mailto:**"
- This allows us to re-classify the issue type as "**Email Attribute Spoofing** " instead of "**Open Redirect**"
  - giving more granularity into the type of risk
  - providing results that make more sense to the user

De-obfuscation

BEFORE

http://www.evite.com/party/invitations/com.evite.neo.gallery.nocache.js

```
1  function com_evite_neo_Gallery(){var l='',F='" for "gwt:onPropertyEr
   <\/script>',p='#',r='/',vb='0C57A0DD1132C7DFA25E35FB70390037.cache.html',tb='350409DE842147FB19385
   <\/script>',ac='<script id="',Ab='<script language="javascript" src="',A='=',q='?',C='Bad handler
   com_evite_neo_Gallery',kb='ie6',ab='iframe',t='img',bb="javascript:'",pb='loadExternalRefs',v='me
   <\/script>',ob='unknown',fb='user.agent',hb='webkit';var cc=window,k=document,bc=cc.__gwtStatsEven
   {try{return cc.external&&(cc.external.gwtOnLoad&&cc.location.search.indexOf(yb)==-1)}catch(a){retu
2  function oc(){if(wc&&mc){var c=k.getElementById(m);var b=c.contentWindow;b.__gwt_initHandlers=com_
3  function jc(){var j,h=Fb,i;k.write(ac+h+n);i=k.getElementById(h);j=i&&i.previousSibling;while(j&&j
4  ;if(j&&j.src){gc=f(j.src)}if(gc==l){var e=k.getElementsByTagName(s);if(e.length>0){gc=e[e.length-1
5  function tc(){var f=document.getElementsByTagName(v);for(var d=0,g=f.length;d<g;++d){var e=f[d],h=
   {alert(C+b+F)}}}}}}
6  function yc(d,e){var a=fc;for(var b=0,c=d.length-1;b<c;++b){a=a[d[b]]||(a[d[b]]=[])}a[d[c]]=e}
7  function ic(d){var e=vc[d](),b=zc[d];if(e in b){return e}var a=[];for(var c in b){a[b[c]]=c}if(uc)
```

http://www.evite.com/party/invitations/com.evite.neo.gallery.nocache.js

```
1  function com_evite_neo_Gallery(){var l='',F='" for "gwt:onPropertyE
   <\/script>',p='#',r='/',vb='0C57A0DD1132C7DFA25E35FB70390037.cache.html',tb='350409DE842147FB1938
   <\/script>',ac='<script id="',Ab='<script language="javascript" src="',A='=',q='?',C='Bad handler
   com_evite_neo_Gallery',kb='ie6',ab='iframe',t='img',bb="javascript:'",pb='loadExternalRefs',v='m
   <\/script>',ob='unknown',fb='user.agent',hb='webkit';var cc=window,k=document,bc=cc.__gwtStatsEve
   {try{return cc.external&&(cc.external.gwtOnLoad&&cc.location.search.indexOf(yb)==-1)}catch(a){ret
2  function oc(){if(wc&&mc){var c=k.getElementById(m);var b=c.contentWindow;b.__gwt_initHandlers=com
3  function jc(){var j,h=Fb,i;k.write(ac+h+n);i=k.getElementById(h);j=i&&i.previousSibling;while(j&&
4  ;if(j&&j.src){gc=f(j.src)}if(gc==l){var e=k.getElementsByTagName(s);if(e.length>0){gc=e[e.length-
5  function tc(){var f=document.getElementsByTagName(v);for(var d=0,g=f.length;d<g;++d){var e=f[d],h
   {alert(C+b+F)}}}}}}
6  function yc(d,e){var a=fc;for(var b=0,c=d.length-1;b<c;++b){a=a[d[b]]||(a[d[b]]=[])}a[d[c]]=e}
7  function ic(d){var e=vc[d](),b=zc[d];if(e in b){return e}var a=[];for(var c in b){a[b[c]]=c}if(uc
8  var kc;function nc(){if(!kc){kc=true;var a=k.createElement(ab);a.src=bb;a.id=m;a.style.cssText=db
9  vc[fb]=function(){var d=navigator.userAgent.toLowerCase();var b=function(a){return parseInt(a[1])
   {if(b(c)>=1008)return mb}return lb}return ob};zc[fb]={gecko:0,gecko1_8:1,ie6:2,opera:3,safari:4};
   (new Date()).getTime(),type:qb});var xc;if(lc()){xc=rb}else{try{yc([mb],sb);yc([ib],tb);yc([lb],u
10 if(k.addEventListener){k.addEventListener(xb,function(){nc();qc()},false)}var rc=setInterval(func
   {__gwt_scriptsLoaded[zb]=true;document.write(Ab+gc+Bb)}k.write(Cb)}
```

- 1 in 5 issues found with JSA were obfuscated
- We support de-obfuscation of JS files
- Negligible overhead

AFTER

http://www.evite.com/party/invitations/com.evite.neo.gallery.nocache.js

```
4                if (e.length > 0) {
                     gc = e[e.length - 1].href
                 } else {
4 ❶                  gc = f(k.location.href)
                 }
             } else if (gc.match(/^\w+:\/\//)) {} else {
                 var g = k.createElement(t);
```

http://www.evite.com/party/invitations/com.evite.neo.gallery.nocache.js

```
3            while (j && j.tagName != o) {
                 j = j.previousSibling
             }
3 ❷          function f(b) {
                 var a = b.lastIndexOf(p);
                 if (a == -1) {
                     a = b.length
                 }
                 var c = b.indexOf(q);
                 if (c == -1) {
                     c = b.length
                 }
                 var d = b.lastIndexOf(r, Math.min(c, a));
3 ❸              return d >= 0 ? b.substring(0, d + 1) : l
             }
4            ;
             if (j && j.src) {
                 gc = f(j.src)
             }
             if (gc == l) {
                 var e = k.getElementsByTagName(s);
                 if (e.length > 0) {
                     gc = e[e.length - 1].href
                 } else {
4 ❹                  gc = f(k.location.href)
                 }
             } else if (gc.match(/^\w+:\/\//)) {} else {
                 var g = k.createElement(t);
...
9                oc()
         };
         jc();
9 ❺      tc();
         bc && bc({
             moduleName: m, subSystem: x, evtGroup: cb, millis: (new Date()).getTime(), type: qb
         });
...
10           });
             if (!__gwt_scriptsLoaded[zb]) {
                 __gwt_scriptsLoaded[zb] = true;
10 ❻            document.write(Ab + gc + Bb)
             }
             k.write(Cb)
         }
```

JSA Evaluation on Real World Websites

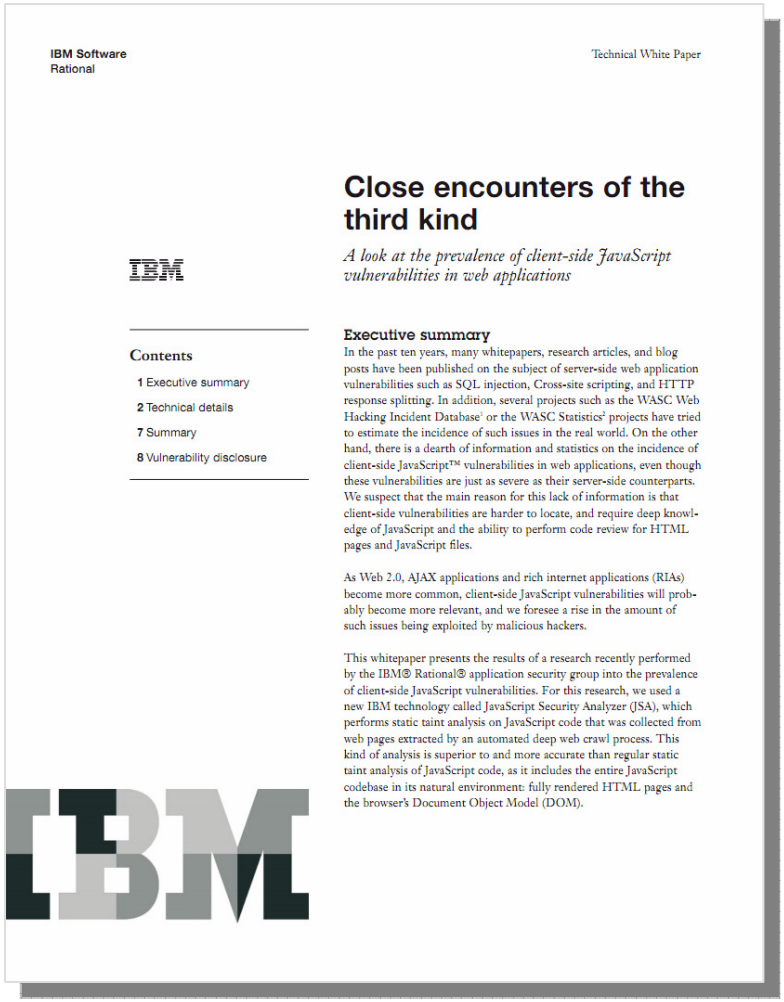
PROCESS

- Scanned 675 real-world websites
  - All Fortune 500
  - Web100 “top websites of 2010”
  - Customer apps
- 200 to 500 pages per site; total >160,000 pages
- Manually reviewed & classified all findings
- Repeatedly scanned and improved JSA based on the results

RESULTS

- >15% sites with confirmed vulnerabilities
  - Primarily DOM-based XSS
- 95% of JSA’s findings are exploitable true positives
  - String Analysis eliminated false positives

Whitepaper with JSA Results



IBM Software
Rational

Technical White Paper

Close encounters of the third kind

A look at the prevalence of client-side JavaScript vulnerabilities in web applications

IBM

Contents

Executive summary

In the past ten years, many whitepapers, research articles, and blog posts have been published on the subject of server-side web application vulnerabilities such as SQL injection, Cross-site scripting, and HTTP response splitting. In addition, several projects such as the WASC Web Hacking Incident Database[1] or the WASC Statistics[2] projects have tried to estimate the incidence of such issues in the real world. On the other hand, there is a dearth of information and statistics on the incidence of client-side JavaScript™ vulnerabilities in web applications, even though these vulnerabilities are just as severe as their server-side counterparts. We suspect that the main reason for this lack of information is that client-side vulnerabilities are harder to locate, and require deep knowledge of JavaScript and the ability to perform code review for HTML pages and JavaScript files.

As Web 2.0, AJAX applications and rich internet applications (RIAs) become more common, client-side JavaScript vulnerabilities will probably become more relevant, and we foresee a rise in the amount of such issues being exploited by malicious hackers.

This whitepaper presents the results of a research recently performed by the IBM® Rational® application security group into the prevalence of client-side JavaScript vulnerabilities. For this research, we used a new IBM technology called JavaScript Security Analyzer (JSA), which performs static taint analysis on JavaScript code that was collected from web pages extracted by an automated deep web crawl process. This kind of analysis is superior to and more accurate than regular static taint analysis of JavaScript code, as it includes the entire JavaScript codebase in its natural environment: fully rendered HTML pages and the browser's Document Object Model (DOM).

IBM

Summary

- What is Hybrid Analysis?
- Correlation
  - Attack Vector Detection
- JavaScript Security Motivation
- Static Analysis overview
- JSA Technology
  - String Analysis
  - De-obfuscation

Smarter security for a smarter planet

www.ibm.com/software/rational

www.ibm.com/software/rational