Exploiting Web 2.0 – Next Generation Vulnerabilities

Shreeraj Shah
Chapter Lead
Founder & Director
Blueinfy Solutions
shreeraj@blueinfy.com

OWASP
EU09 Poland

The OWASP Foundation
http://www.owasp.org

Who Am I?

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

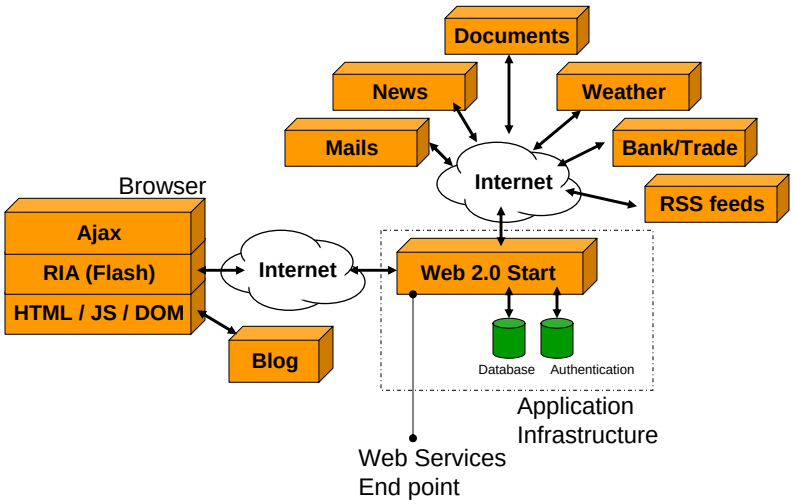
- **Founder & Director**
  - Blueinfy Solutions Pvt. Ltd.
  - SecurityExposure.com
- **Past experience**
  - Net Square, Chase, IBM & Foundstone
- **Interest**
  - Web security research
- **Published research**
  - Articles / Papers – Securityfocus, O'erilly, DevX, InformIT etc.
  - Tools – wsScanner, 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

Real Case Study

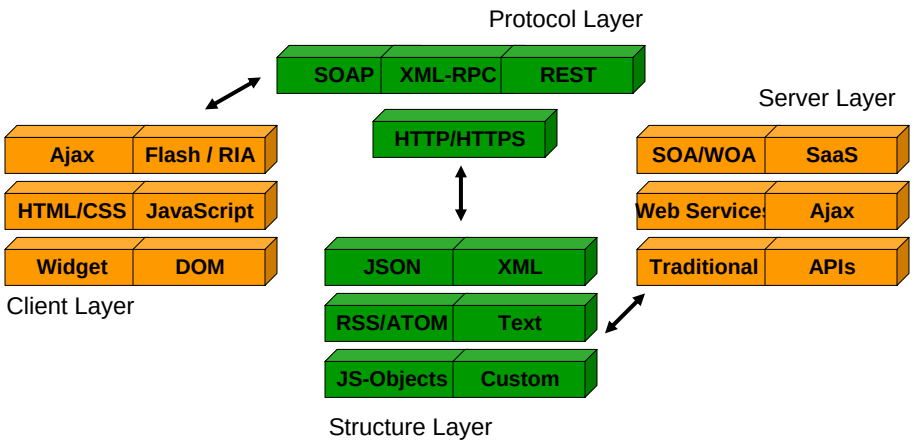
- Web 2.0 Portal – Buy / Sell
- Technologies & Components – Dojo, Ajax, XML Services, Blog, Widgets
- Scan with tools/products **failed**
- Security issues and hacks
  - SQL injection over XML
  - Ajax driven XSS
  - Several XSS with Blog component
  - Several information leaks through JSON fuzzing
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

Web 2.0 Architecture and Security

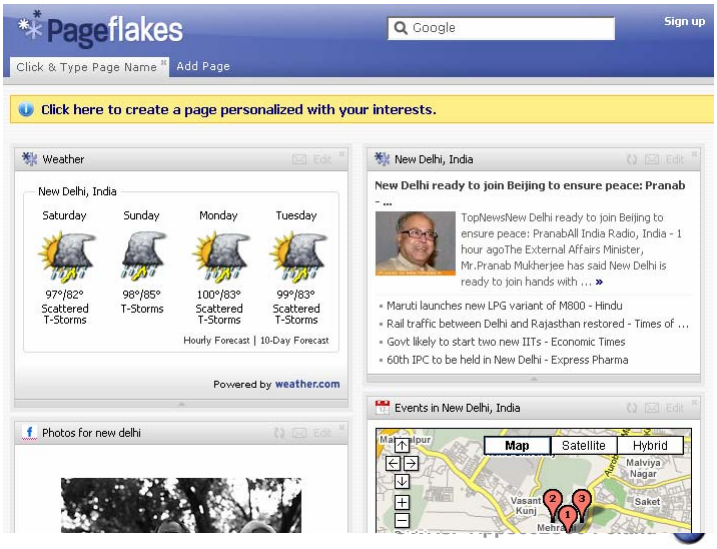
Web 2.0 Architecture



Web 2.0 Components

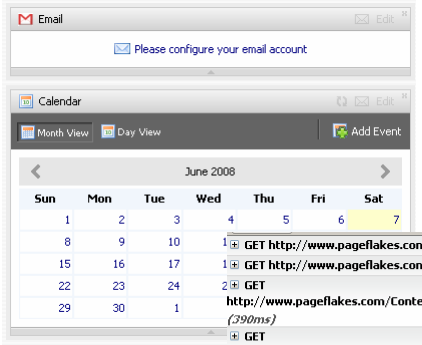


Case study - Pageflakes



Case study - Pageflakes

Widgets



Web Services

Impact of Web 2.0

- Application Infrastructure

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

Impact of Web 2.0

- Security Threats

Changing dimension	Web 1.0	Web 2.0
***(T1)* Entry points**	Structured	Scattered and multiple
***(T2)* Dependencies**	Limited	• Multiple technologies • Information sources • Protocols
***(T3)* Vulnerabilities**	Server side [Typical injections]	• Web services [Payloads] • Client side [XSS & XSRF]
***(T4)* Exploitation**	Server side exploitation	Both server and client side exploitation

Changes in approach

- Methodology

Changing dimension	Web 1.0	Web 2.0
Footprinting	Typical with "Host" and DNS	Empowered with search
Discovery	Simple	Difficult with hidden calls
Enumeration	Structured	Several streams
Scanning	Structured and simple	Difficult with extensive Ajax
Automated attacks	Easy after discovery	Difficult with Ajax and web services
Reverse engineering	On the server-side [Difficult]	Client-side with Ajax & Flash
Code reviews	Focus on server-side only	Client-side analysis needed

Web 2.0 Security

- 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

Web 2.0 Security

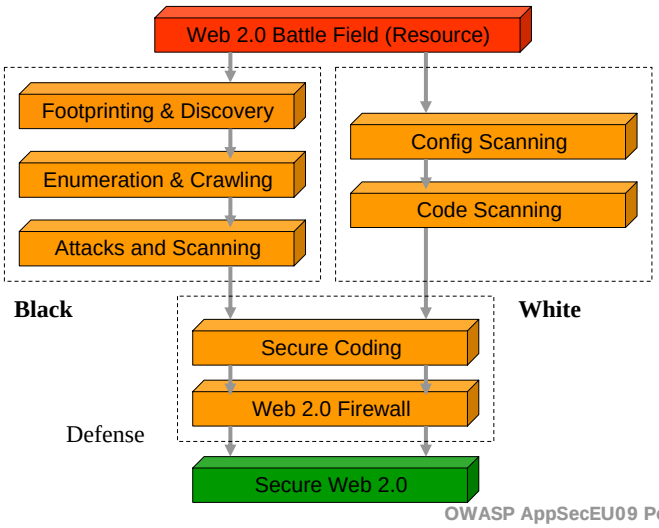
- Mashup and un-trusted sources
- RSS feeds manipulation and its integration
- Single Sign On and information convergence at one point
- Widgets and third-party components are bringing security concerns
- Old attacks with new carriers

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

Web 2.0 – Methodologies & Challenges

Methodology, Scan and Attacks



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

Web 2.0 Fingerprinting, Discovery & Crawling

Application Server Fingerprinting

- Identifying Web and Application servers.
- Forcing handlers to derive internal plugin or application servers like Tomcat or WebLogic.
- Looking for Axis or any other Web Services container.
- Gives overall idea about infrastructure.

Fingerprinting

- 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.

Discovery

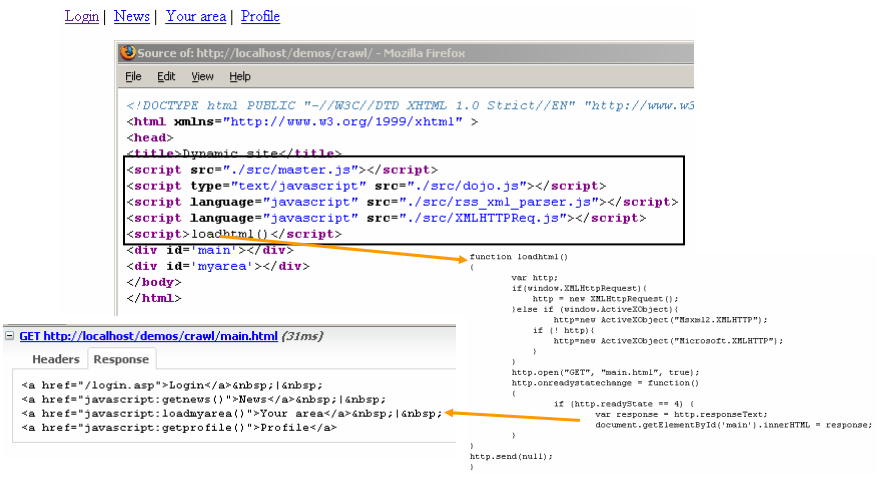
- Ajax running with various different structures.
- Developers are adding various different calls and methods for it.
- JavaScript can talk with back end sources.
- Mashups application talking with various sources.
- It has significant security impact.
- JSON, Array, JS-Object etc.
- Identifying and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

Discovery

Crawling challenges

- 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.

Ajax driven site



Crawling with Ruby/Watir

```
require 'watir'
include Watir
ie=IE.new
ie.goto("http://localhost/demos/crawl/")
ie.show_links
ie.links[2].click
ie.show_links
ie.links[3].click
ie.show_links
ie.links[4].click
ie.show_links
```

```
Command Prompt

D:\scanweb2.0>crawl.rb
index name          id          href
          text/src
1         Login               http://localhost/login.asp
2         News                javascript:getnews()
3         Your area           javascript:loadmyarea()
4         Profile             javascript:getprofile()
index name          id          href
          text/src
1         Login               http://localhost/login.asp
2         News                javascript:getnews()
3         Your area           javascript:loadmyarea()
4         Profile             javascript:getprofile()
index name          id          href
          text/src
1                             http://localhost/login.asp
```


Web 2.0 Vulnerabilities & Exploits

Ajax code review

- Ajax scripts are on client side and important to do source sifting on it
- Looking for business logic and vulnerabilities on Ajax components
- JavaScript analysis and review
- Looking for malicious calls and pattern of malware if any
- Very sensitive in mashup context
- In browser debugging would be very handy

Source Code Disclosure

- Hidden Ajax calls fetching files
- Discovering those calls
- Exploiting with ../ and getting files
- Common with content managements systems
- RIA calls can be discovered as well

SQL 2.0

- 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

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

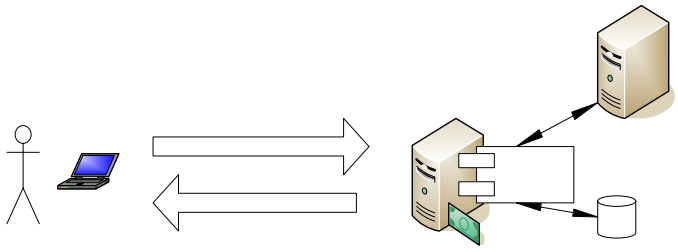
Cross Site Request Forgery (CSRF)

- Generic CSRF is with GET / POST
- Forcefully sending request to the target application with cookie replay
- Leveraging tags like
  - IMG
  - SCRIPT
  - IFRAME
- Not abide by SOP or Cross Domain is possible

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

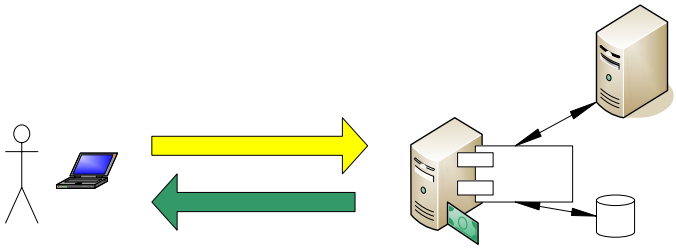
One Way CSRF Scenario



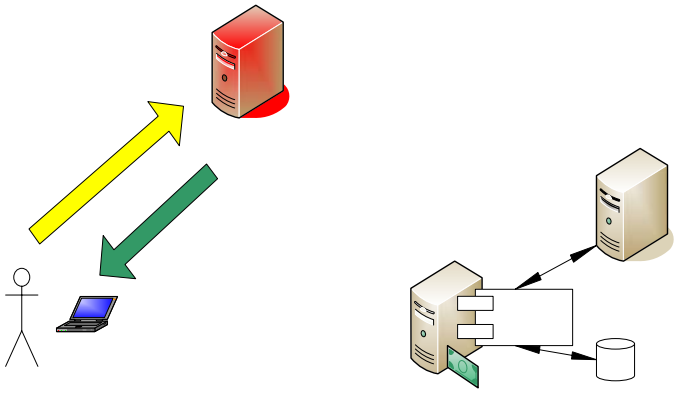
One Way CSRF Scenario

Login request

Authentication / Cookie



One Way CSRF Scenario

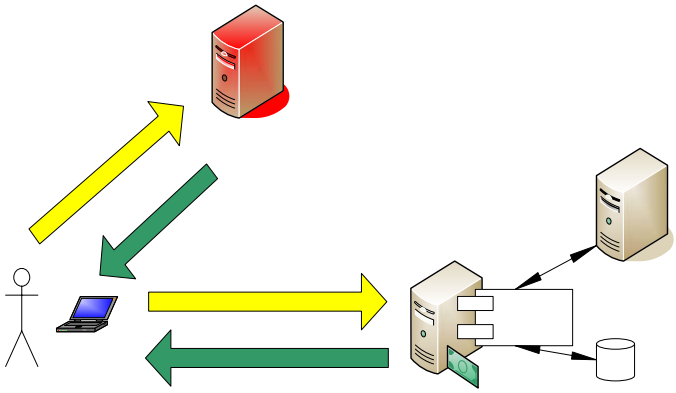


ɔcker)

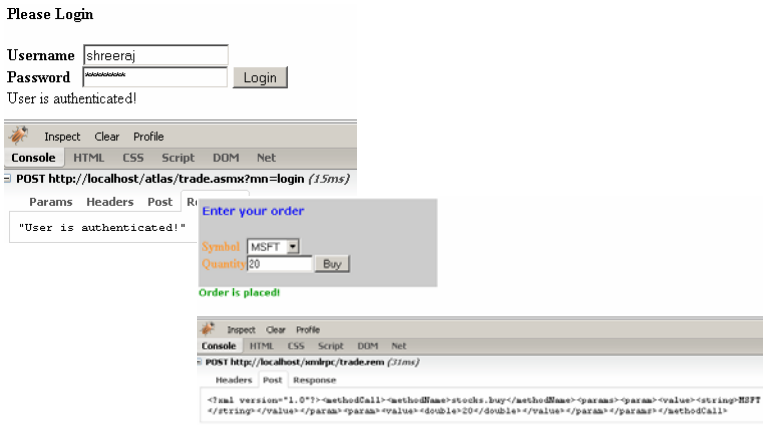
Attacker's Site

page

One Way CSRF Scenario



One-Way CSRF



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></value></param><param><value><double>26</double></value></param></params></methodCall>'>
- </FORM>
- <script>document.buy.submit();</script>
- </body>
- </html>

Forcing XML

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

Cross Site Scripting (XSS)

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

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.

DOM

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

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);
}
```


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();
```

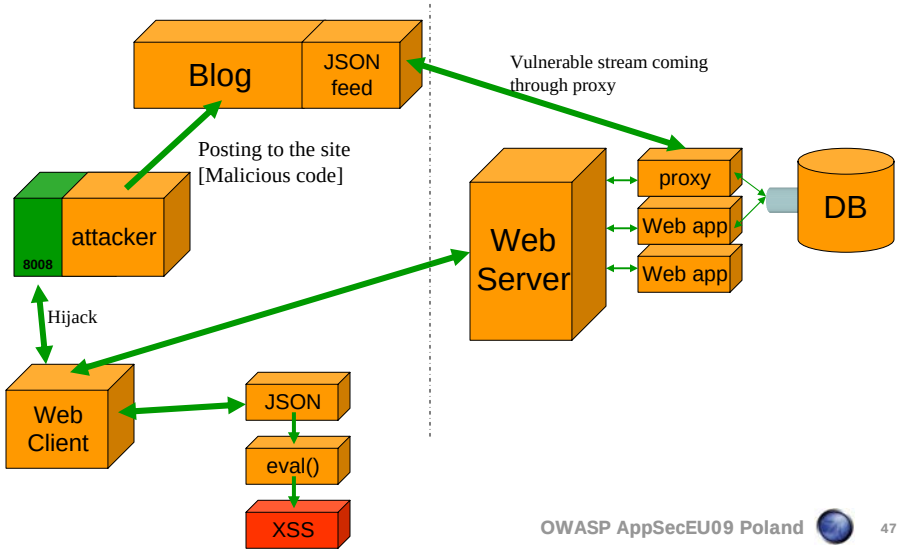
DOM based XSS

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(...)

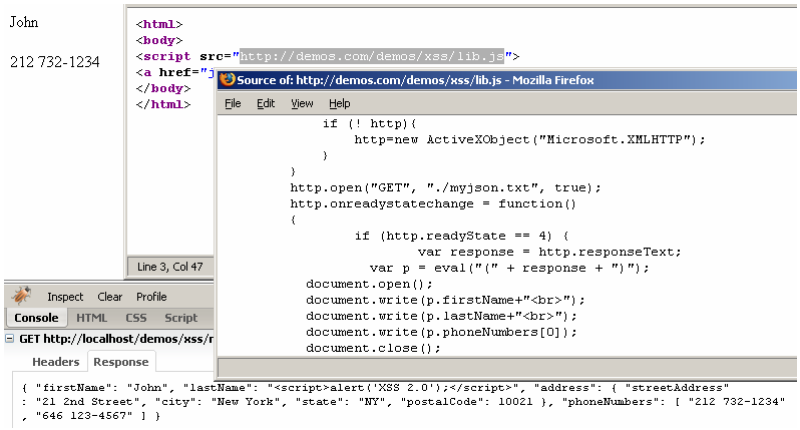
DOM based XSS

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

Scenario



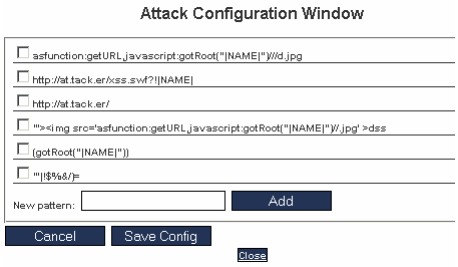
XSS with JSON stream



XSS with RIA

- Applications running with Flash components
- getURL – injection is possible
- SWFIntruder
- Flasm/Flare

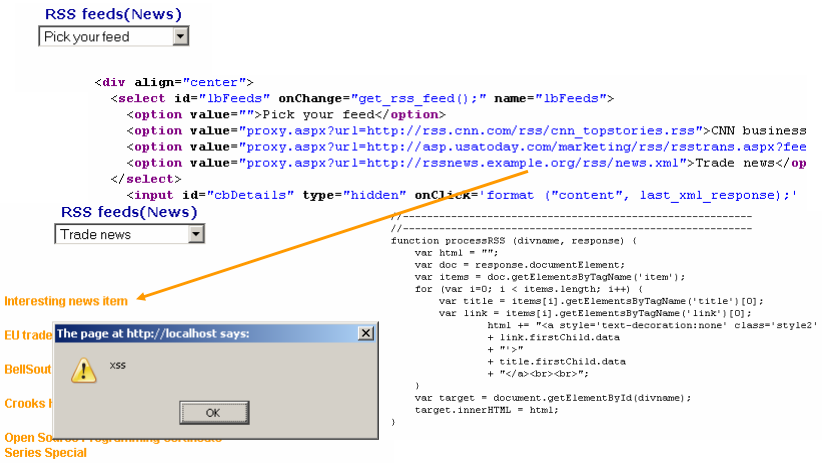
(http://www.nowrap.de/)



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.

RSS feeds



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

SOA Hacks

- Discovering Web Services
- Profiling and Enumerating through WSDL
- Attacking Web Services
- SOAP manipulation is the key

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

Conclusion – Questions...