Who Am I?

- **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

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

![Diagram of Web 2.0 Architecture with components such as Browser, Ajax, RIA (Flash), HTML / JS / DOM, Blog, Document, News, Weather, Mails, Bank / Trade, RSS feeds, Internet, Web 2.0 Start, Database, Authentication, Application Infrastructure, Web Services End point, and Client Layer, Protocol Layer, Server Layer, and Structure Layer.]

Web 2.0 Components

![Diagram of Web 2.0 Components with components such as Ajax, Flash / RIA, HTML / CSS, JavaScript, Widget, DOM, SOAP, XML-RPC, REST, HTTP / HTTPS, JSON, XML, RSS / ATOM, Text, JS-Objects, Custom, and layers such as Client Layer, Protocol Layer, Server Layer, and Structure Layer.]
Case study - Pageflakes

Widgets

Web Services

GET http://www.pageflakes.com/WS/services/user/GetUser

GET http://www.pageflakes.com/WS/services/GetUserServices

GET http://www.pageflakes.com/WS/services/GetMap

POST http://www.pageflakes.com/WS/services/PostMap
### Impact of Web 2.0

**Application Infrastructure**

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

### Impact of Web 2.0

**Security Threats**

<table>
<thead>
<tr>
<th>Changing dimension</th>
<th>Web 1.0</th>
<th>Web 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T1) Entry points</td>
<td>Structured</td>
<td>Scattered and multiple</td>
</tr>
<tr>
<td>(T2) Dependencies</td>
<td>Limited</td>
<td>• Multiple technologies</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Information sources</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Protocols</td>
</tr>
<tr>
<td>(T3) Vulnerabilities</td>
<td>Server side [Typical injections]</td>
<td>• Web services [Payloads]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Client side [XSS &amp; XSRF]</td>
</tr>
<tr>
<td>(T4) Exploitation</td>
<td>Server side exploitation</td>
<td>Both server and client side exploitation</td>
</tr>
</tbody>
</table>
Changes in approach

- Methodology

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

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

- Footprinting & Discovery
- Enumeration & Crawling
- Attacks and Scanning
- Config Scanning
- Code Scanning
- Secure Coding
- Web 2.0 Firewall
- Secure Web 2.0
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.
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

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

[Diagram showing a one way CSRF scenario with a login request and authentication/cookie]
One Way CSRF Scenario

Attacker

One Way CSRF Scenario

Attacker's Site
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='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>
  </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

```javascript
function getajax()
{
  var http;
  if(window.XMLHttpRequest)
  {
    http = new XMLHttpRequest();
  }
  else if (window.ActiveXObject)
  {
    http=new ActiveXObject("Msoxml2.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

```javascript
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.createElement(...)  
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

Web Client

Post to the site [Malicious code]

Web Server

Hijack

XSS

JSON

eval()

Posting to the site

Vulnerable stream coming through proxy

Web app

proxy

DB

Web app

Scenario Blog

JSON feed

OWASP AppSecEU09 Poland

XSS with JSON stream

John

212 732-1234

<script src="http://demo05.dmv.com/demos/xss/th.js"></script>

Source of: http://demo05.dmv.com/demos/xss/th.js

/* Source of: http://demo05.dmv.com/demos/xss/th.js

var r= "";
for (var i=0; i<10; i++) {
    r += JSON.stringify({"name": i,"value": i});
}

console.log(r);

*/

GET http://localhost/demos/xss/th.js

OWASP AppSecEU09 Poland
**XSS with RIA**

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

(https://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

**RSS feeds (News)**

| Pick your feed |

```html
<select id='lbFoods' onChange='get_rss_feed();' name='lbFoods'>
  <option value=''>Pick your feed</option>
  <option value='http://zoe.com/com/zoe/com_topstories.rss'>OW2 business</option>
  <option value='http://tonnies.develop.org/rss/news.xml'>Trade news</option>
</select>
```

**RSS feeds (News)**

| Trade news |

---

**Interesting news from**

**EU trade**
The page at http://ec.europa.eu

**Blog**
RSS

**Crosstab**

Open Source Programming Conferences
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

SOA Hacks

- Discovering Web Services
- Profiling and Enumerating through WSDL
- Attacking Web Services
- SOAP manipulation is the key
Conclusion – Questions...