Hacking Web 2.0
Art and Science of Vulnerability Detection

Shreeraj Shah
Pune, India
Who am I?

• Founder & Director
  – Blueinfy Solutions Pvt. Ltd. (Brief)
• 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, wsChess etc.
  – Advisories - .Net, Java servers etc.
• Books (Author)
  – Hacking Web Services (Thomson 2006)
  – Web Hacking (AWL 2003)
  – Web 2.0 Security (Work in progress)

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

- Web 2.0 overview and security concerns
- Ajax Security – Attacks and Defense
  - Methods
  - Vectors
  - Defense
- Web Services – Attacks and Defense
  - Methodology
  - Assessment and Tools
  - Defense
Web 2.0 Trends

• 80% of companies are investing in Web Services as part of their Web 2.0 initiative (McKinsey2007 Global Survey)

• By the end of 2007, 30 percent of large companies will have some kind of Web 2.0-based business initiative up and running. (Gartner)

• 2008. Web Services or Service-Oriented Architecture (SOA) would surge ahead. (Gartner)
Web 2.0 – Ajax & Web Services

Browser

- Ajax
- RIA (Flash)
- HTML / JS / DOM

Internet

Web Services

- Local Application
  - Database
  - Authentication

Documents

News

Weather

Emails

Bank/Trade

RSS feeds

Blog
Web 2.0 Layers

Browser
- Ajax
- HTML/CSS
- Widget

Structures
- Flash / RIA
- JavaScript
- DOM
- XML
- JSON

Protocols
- JSON-RPC
- REST
- XML-RPC
- SOAP

Server-Side
- Services
- SaaS
- Open APIs

HTTP(S)
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
Ajax Security – Attacks & Defense

- Basics
- Structures and streams
- Fingerprinting
- Scanning and Enumeration
- XSS and CSRF issues
- Securing code base
Ajax basics

• Asynchronous JavaScript and XML
function loadhtml()
{
    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", "main.html", true);
    http.onreadystatechange = function() {
        if (http.readyState == 4) {
            var response = http.responseText;
            document.getElementById('main').innerHTML = response;
        }
    }
    http.send(null);
}
Ajax & Data structures

- Ajax is using various data streams
- Developers are innovating this field
- JavaScript can talk with back end sources
- Mashups application can be leveraged
- It is important to understand these streams
- It has significant security impact
- JSON, Array, JS-Object etc.
Cross-domain calls

- Browser security doesn’t support cross domain calls
- But cross domain callback with JavaScript is possible
- This can be lethal attack since cross domain information get executed on the current DOM context.
- Developers put proxy to bypass the SOP.
Ajax fingerprinting

- Determining Ajax calls
- Framework fingerprinting
- Running with what?
  - Atlas
  - GWT
  - Etc.
- Ajaxfinger a tool to achieve this
- Can help in assessment process
- RIA finger printing is possible
Ajax attack points

- Ajax components & Widgets
- Cross domain vulnerable browsers and callback implementations
- DOM manipulation calls and points
- Insecure eval()
- HTML tags
- Intranet nodes and internal resources
Ajax attack vectors

- Entry point scanning and enumeration
- Cross site scripting (XSS) attacks
- Cross site Request Forgery (CSRF) issues
- Client side code reverse engineering
- Security control and validation bypassing
- Local privacy information enumeration
- Ajax framework exploitation – known bugs
Ajax Scanning

- Scanning Ajax components
- Retrieving all JS include files
  - Part of <SCRIPT SRC=….> 
- Identifying XHR calls 
- Grabbing function 
- Mapping function to DOM event 
- Scanning code for XSS – look for eval() and document.write()
Ajax serialization issues

- Ajax processing various information coming from server and third party sources. – XSS opportunities

```javascript
message = {
    from : "john@example.com",
    to : "jerry@victim.com",
    subject : "I am fine",
    body : "Long message here",
    showsubject : function(){document.write(this.subject)}
};
```

XSS
Ajax serialization issues

- JSON issues

```json
{"bookmarks":[{"Link":"www.example.com","Desc":"Interesting link"}]}
```

- JS – Array manipulation

```javascript
new Array(“Laptop”, “Thinkpad”, “T60”, “Used”, “900$”, “It is great and I have used it for 2 years”)
```
Ajax and JS manipulation

- JavaScript exploitation – XSS
- Identifying DOM points like `document.write()`
- `Eval()` – another interesting point
- Attack APIs and tools for exploitation
- Lot can be done by an attacker from session hijacking to key loggers
Ajax and RSS injection

- RSS feeds are another entry point to the browser
- Injecting script to the RSS feeds and Ajax call may execute it.
- One click – Malformed linked injected into it and can lead to exploit “javascript:”
- Leveraging events – onClick, onMouse etc.
Ajax Crawling

• Crawling Ajax driven app – a challenge
• Resources are hidden in JavaScript
• Simple scanner will fail
• Crawling with actual DOM context
• Automated crawling with browser is required
• How?
Defending Ajax

- No business logic information on client side.
- Do not trust third party source – filter it out
- No direct cross domain call back
- Filtering at browser level before processing information
- Avoiding client side validation
Defending Ajax

- No secret in Ajax calls
- Proper data structure selection and frameworks
- Avoid client side validation
- Securing client side calls like eval() and document.write()
- HTML tags filtering before serving to end client
Web Services – Attacks & Defense

• Methodology
• Footprinting & Discovery
• Profiling and Enumeration
• Scanning and Fuzzing
• Attack vectors
• Scanning code for vulnerabilities
• Defense by filtering
Methodology

Blackbox

Insecure Web Services

Footprinting & Discovery

Enumeration & Profiling

Vulnerability Detection

Whitebox

Code / Config Scanning

Secure Coding

Web Services Firewall

Secure Web Services

Defense & Countermeasure

Blueinfy
Footprinting and Discovery

• Objective: Discovering Web Services running on application domain.

• Methods
  – Primary discovery
    • Crawling and spidering
    • Script analysis and page scrubbing
    • Traffic analysis
  – Secondary discovery
    • Search engine queries
    • UDDI scanning
• Crawling the application and mapping file extensions and directory structures, like ".asmx"
• Page scrubbing – scanning for paths and resources in the pages, like atlas back end call to Web Services.
• Recording traffic while browsing and spidering, look for XML based traffic – leads to XML-RPC, REST, SOAP, JSON calls.
Primary Discovery - Demos

- Page scanning with grep – Look in JavaScripts for URLs, Paths etc.
- Crawling – Simple!
- Scanning for Atlas references – Framework creates stubs and proxy. – scanweb2.0.scanatlas
- Urlgrep can be used as well.
Secondary Discovery

• Searching UDDI server for Web Services running on particular domain.
  – Three tactics for it – business, services or tModel.

• Running queries against search engines like Google or MSN with extra directives like “inurl” or “filetype”
  – Look for “asmx”

• wsScanner – Discovery!
Enumerating and Profiling

• Scanning WSDL
  – Looking for Methods
  – Collecting In/Out parameters
  – Security implementations
  – Binding points
  – Method signature mapping
Scanning strategies

• Manual invocation and response analysis.
• Dynamic proxy creation and scanning.
• Auto auditing for various vectors.
• Fuzzing Web Services streams – XML or JSON

• Response analysis is the key
  – Look for fault code nodes
  – Enumerating fault strings
  – Dissecting XML message and finding bits
  – Hidden error messages in JSON
Cross Site Scripting (XSS)

- XSS is possible through Web Services.
- It would be DOM based XSS via `eval()`.
- JSON-RPC based stream coming in the browser and get injected into DOM.
- Source of stream can be of third party and Un-trusted.
- XML streams coming in the browser and can cause XSS via `document.write` call.
Injection Flaws

• Web Services methods are consuming parameters coming from end users.
• It is possible to inject malicious characters into the stream.
• It can break Web Services code and send faultstring back to an attacker
• Various injections possible – SQL and XPATH
Malicious File Execution

- Malicious command can be injected through the parameter.
- WS supports attachments as well and that can lead to uploading a file.
- This can give remote command execution capability to the attacker.
Insecure Direct Object Reference

• Injecting characters to break file system sequences.
• Faultcode spits out internal information if not protected.
• Customized error shows the file references.
• Access to internal file and full traversal to directories
• Inspecting methods and parameters in the profile stage can help.
Cross Site Request Forgery

- CSRF with XML streams
- XML-RPC or SOAP based request can be generated from browsers.
- Splitting form and XML injection is possible – interesting trick.
- If Content-Type is not validated on the server then it can cause a potential CSRF.
- XForms usage in browser can produce XML requests to attack CSRF.
Code Analysis for Web Services

• Scanning the code base.
• Identifying linkages.
• Method signatures and inputs.
• Looking for various patterns for SQL, LDAP, XPATH, File access etc.
• Checking validation on them.
• Code walking and tracing the base - Key
Code filtering with IHttpModule

- Regular firewall will not work
- Content filtering on HTTP will not work either since it is SOAP over HTTP/HTTPS
- SOAP level filtering and monitoring would require
- ISAPI level filtering is essential
- SOAP content filtering through IHttpModule
HTTP Stack for .Net

- HttpRuntime
- HttpApplicationFactory
- HttpApplication
- IHttpModule
- HttpHandlerFactory
- Handler

Web Application Firewall & IDS
IHTTPModule for Web Services Firewall

- Code walkthrough – Events and Hooks
- Loading the DLL
- Setting up the rules
- Up and running!
- Demo.
Thanks!

• Questions?

  – shreeraj@blueinfy.com