Web 2.0 Security Part 4: A MASHup of Problems

Lori MacVittie, 2007-18-07

This is Part 4 of a series on Web 2.0 Security.

A good way to remember things is to use mnemonics, so when you're trying to list the security issues relevant to Web 2.0 just remember this: it's a **MASH**up.

- **M**ore of everything.
- **A**symmetric data formats
- **S**cripting based
- **H**idden URLs and code

This episode is brought to you by the letter "H".

Hidden URLs

AJAX and Web 2.0 works because of the use of the XMLHTTPRequest object via JavaScript to invoke remote calls on the server for the purposes of retrieving data. It still requires a valid URL, just like an HREF, but the URLs are often "hidden" in the JavaScript rather than in the attribute tags of the appropriate HTML elements.

Worse, these URLs are often crafted at runtime based on a number of parameters determined by the developer.

While this is great for stopping automated crawlers and bots from getting at those URLs, it makes it difficult to implement automated testing of the application because most web-based vulnerability assessment and testing tools aren't capable of evaluating JavaScript and recreating those URLs.

The hidden URL issue also makes it difficult to implement a comprehensive security policy for the application that might make use of those URLs - such as an application specific policy that only allows calls to known or documented URLs. The referrer cannot always be trusted as it is often either not set, or can be set by the developer.

...and code

Perhaps even more dangerous is the fact that the code executing AJAX calls is often hidden within a toolkit. While toolkits like Dojo and Script.aculo.us make development of Web 2.0 applications a breeze, they too often become black boxes even though they are JavaScript libraries *and* open source, which makes them easily perusable by anyone who wishes to examine them. While many open source projects now include at least MD5 hashes to verify that the files downloaded are equivalent to the ones offered, AJAX toolkits have yet to provide any assurance that the toolkit downloaded is actually the one intended to be downloaded.

Also hidden from sight are extensions to core JavaScript objects like the XMLHTTPRequest object. JavaScript is highly extensible, just feed it some code and evaluate and it has new - and possibly nefarious - functionality. In the same way that viruses and malware are silently loaded onto victims' machines via WebSites, so too could malicious JavaScript code be loaded into pages and applications - without the user knowing it happened.

The Risks

- There is no way to distinguish malicious script from valid script, leaving attackers free to inject scripts into the client via infected web sites or other techniques that modify the core behavior of Web 2.0 applications
- Determining which URLs (scripts) are valid for an application require intense scrutiny of the code or documentation from developers, both of which are time consuming and are rarely complete enough for a comprehensive security policy to be implemented
- Malicious code could be hidden in toolkits before the toolkit is downloaded
- Phishing-like mechanisms could be used to extend JavaScript functionality without the user aware that it has occurred, which can lead to theft of cookies or other sensitive data.

Our next (and final) chapter in this Web 2.0 Security series will be examining the strategies to secure Web 2.0 applications.

Imbibing: Apple Juice

Technorati tags: F5, MacVittie, application security, Web 2.0, AJAX, security

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com