Are AJAX Applications Vulnerable to Hack Attacks?

The importance of Securing AJAX Web Applications

This paper reviews AJAX technologies with specific reference to JavaScript and briefly documents the kinds of vulnerability classes that should raise security concerns among developers, website owners and the respective visitors. The proposed solution suggests auditing AJAX and JavaScript based applications with a web vulnerability scanner that not only parses the HTML code of a webpage to identify embedded JavaScript, but also executes the code. Automating the process is also key when considering the increasing complexity of such web applications.

Table of Contents

1. AJAX and JavaScript

1.1 The Technologies Explained

Fuelled by the increased interest in Web 2.0, AJAX (Asynchronous JavaScript Technology and XML) is attracting the attention of businesses all round the globe.

One of the main reasons for the increasing popularity of AJAX is the scripting language used – JavaScript (JS) which allows: dynamic forms to include built-in error checking, calculation areas on pages, user interaction for warnings and getting confirmations, dynamically changing background and text colours or "buttons", reading URL history and taking actions based on it, open and control windows, providing different documents or parts based on user request (i.e., framed vs. non-framed).

AJAX is not a technology; rather, it is a collection of technologies each providing robust foundations when designing and developing web applications:

- **XHTML or HTML and Cascading Style Sheets (CSS)** providing the standards for representing content to the user.
- **Document Object Model (DOM)** that provides the structure to allow for the dynamic representation of content and related interaction. The DOM exposes powerful ways for users to access and manipulate elements within any document.
- **XML and XSLT** that provide the formats for data to be manipulated, transferred and exchanged between server and client.
- **XML HTTP Request:** The main disadvantages of building web applications is that once a particular webpage is loaded within the user's browser, the related server connection is cut off. Further browsing (even) within the page itself requires establishing another connection with the server and sending the whole page back even though the user might have simply wanted to expand a simple link. XML HTTP Request allows asynchronous data retrieval or ensuring that the page does not reload in its entirety each time the user requests the smallest of changes.
- **JavaScript (JS)** is the scripting language that unifies these elements to operate effectively together and therefore takes a most significant role in web applications.

As such, AJAX is meant to increase interactivity, speed, and usability. The technologies have prompted a richer and friendly experience for the user as web applications are designed to imitate 'traditional' desktop applications including Google Docs and Spreadsheets, Google Maps and Yahoo! Mail.

At the start of a web session, instead of loading the requested webpage, an AJAX engine written in JS is loaded. Acting as a "middleman", this engine resides between the user and the web server acting both as a rendering interface and as a means of communication between the client browser and server.

The difference which this functionality brings about is instantly noticeable. When sending a request to a web server, one notices that individual components of the page are updated independently (asynchronous) doing away with the previous need to wait for a whole page to become active until it is loaded (synchronous).

Imagine webmail – previously, reading email involved a variety of clicks and the sending and retrieving of the various frames that made up the interface just to allow the presentation of the various emails of the user. This drastically slowed down the user's experience. With asynchronous transfer, the AJAX application completely eliminates the "start-stop-start-stop" nature of interaction on the web – requests to the server are completely transparent to the user.

Another noticeable benefit is the relatively faster loading of the various components of the site which was requested. This also leads to a significant reduction in bandwidth required per request since the web page does not need to reload its complete content.

Other important benefits brought about by AJAX coded applications include: insertion and/or deletion of records, submission of web forms, fetching search queries, and editing category trees - performed more effectively and efficiently without the need to request the full HTML of the page each time.

1.2 The Vulnerabilities

1.2.1 AJAX Vulnerabilities

Although a most powerful set of technologies, developers must be aware of the potential security holes and breeches to which AJAX applications have (and will) become vulnerable.

According to Pete Lindstrom, Director of Security Strategies with the Hurwitz Group, Web applications are the most vulnerable elements of an organization's IT infrastructure today. An increasing number of organizations (both for-profit and not-for-profit) depend on Internet-based applications that leverage the power of AJAX. As this group of technologies becomes more complex to allow the depth and functionality discussed, and, if organizations do not secure their web applications, then security risks will only increase.

Increased interactivity within a web application means an increase of XML, text, and general HTML network traffic. This leads to exposing back-end applications which might have not been previously vulnerable, or, if there is insufficient server-side protection, to giving unauthenticated users the possibility of manipulating their privilege configurations.

There is the general misconception that in AJAX applications are more secure because it is thought that a user cannot access the server-side script without the rendered user interface (the AJAX based webpage). XML HTTP Request based web applications obscure server-side scripts, and this obscurity gives website developers and owners a false sense of security – obscurity is not security. Since XML HTTP requests function by using the same protocol as all else on the web (HTTP), technically speaking, AJAX-based web applications are vulnerable to the same hacking methodologies as 'normal' applications.

Subsequently, there is an increase in session management vulnerabilities and a greater risk of hackers gaining access to the many hidden URLs which are necessary for AJAX requests to be processed.

Another weakness of AJAX is the process that formulates server requests. The Ajax engine uses JS to capture the user commands and to transform them into function calls. Such function calls are sent in plain visible text to the server and may easily reveal database table fields such as valid product and user IDs, or even important variable names, valid data types or ranges, and any other parameters which may be manipulated by a hacker.

With this information, a hacker can easily use AJAX functions without the intended interface by crafting specific HTTP requests directly to the server. In case of cross-site scripting, maliciously injected scripts can actually leverage the AJAX provided functionalities to act on behalf of the user thereby tricking the user with the ultimate aim of redirecting his browsing session (e.g., phishing) or monitoring his traffic.

1.2.2 JavaScript Vulnerabilities

Although many websites attribute their interactive features to JS, the widespread use of such technology brings about several grave security concerns.

In the past, most of these security issues arose from worms either targeting mailing systems or exploiting Cross Site Scripting (XSS) weaknesses of vulnerable websites. Such self-propagating worms enabled code to be injected into websites with the aim of being parsed and/or executed by Web browsers or e-mail clients to manipulate or simply retrieve user data.

As web-browsers and their technological capabilities continue to evolve, so does malicious use reinforcing the old and creating new security concerns related to JS and AJAX. This technological advancement is also occurring at a time when there is a significant shift in the ultimate goal of the hacker whose primary goal has changed from acts of vandalism (e.g., website defacement) to theft of corporate data (e.g., customer credit card details) that yield lucrative returns on the black market.

XSS worms will become increasingly intelligent and highly capable of carrying out dilapidating attacks such as widespread network denial of service attacks, spamming and mail attacks, and rampant browser exploits. It has also been recently discovered that it is possible to use JS to map domestic and corporate networks, which instantly makes any devices on the network (print servers, routers, storage devices) vulnerable to attacks.

Ultimately such sophisticated attacks could lead to pinpointing specific network assets to embed malicious JS within a webpage on the corporate intranet, or any AJAX application available for public use and returning data.

The problem to date is that most web scanning tools available encounter serious problems auditing web pages with embedded JS. For example, client-side JS require a great degree of manual intervention (rather than automation).

2. The JavaScript Engine of Acunetix WVS

Acunetix WVS version 4.0 is equipped with the Acunetix Client Script Analyzer (CSA) which is a fully automated JS parsing engine that overcomes the general need for manually crawling and following JS links.

To execute its vulnerability audits, Acunetix WVS simulates the manual intervention of a penetration tester (ethical hacking) by first “crawling” the website and web applications identifying its directory structure.

Any standard web browser has a JS engine that interprets and executes (client-side) any JS embedded in HTML pages. Acunetix CSA works in the same manner as such a JS engine.

When a document is loaded within a web browser, HTML tags are parsed to render visible the various elements of the page to the user. At the same time, any JS are executed by the JS engine within the browser allowing events and behaviours to become active within the page presented to the user. Some of these events and behaviours require user intervention, whereas others do not (i.e., implicit scripts or scripts which execute without user intervention).

Similarly, the Acunetix CSA will load a page parsing the HTML code contained therein and executing all events and behaviours found on the crawled page. Acunetix CSA will simulate almost all interfaces which Microsoft's Internet Explorer exposes to its JS engine, and this gives it almost the same automated capabilities that any web browser has.

The main advantage of Acunetix WVS is that whereas most web-application scanners stop at parsing the JS on the webpage, the CSA will actually execute the scripts. Such execution is critical since it allows greater and more in-depth vulnerability checks on the identified JS and AJAX web applications. Thus, Acunetix WVS is able to find the many weaknesses that lie deep in the code of web pages.

The following is a simple HTML example of the sequence of events as processed by the Acunetix CSA:

```
<HTML>
        <head>
              <script>
                 function DoOnLoad
                  {
                   ... // some javascript code which will be executed when the page loads
                  }
              </script>                                                    1stscript
        </head>
        <body onload="DoOnLoad()">
           <p>A paragraf</p>

              <script>
                   document.write("<a href='/somefile.cgi'>Click to do something </a>");
              </script>                                                    2nd script

        <img scr='button.gif' onclick='location.href="/otherfile.cgi"'/>
        </body>
        </HTML>
```

First the Acunetix CSA will parse the HTML code to construct the DOM representation of the page yielding a skeletal structure of the page that shows the sequence in which the tags must be read and executed.

CSA will identify the exact location of all various scripts contained therein and the relevant references to elements, events and behaviours. CSA will execute all the scripts found on the various pages – both those that are activated through user intervention (e.g., OnClick, OnChange) and those that are executed by the system (e.g., OnLoad, OnUnLoad). Acunetix CSA will execute implicit scripts during HTML parsing since these have an effect on the parsing itself (because these scripts actually determine the structure of the page).

```
HTML
    |
    +- HEAD
    |
    +- BODY
       |
       +- P
       |
       +- A
       |
       +- IMG
```

After parsing the page, CSA will activate all the remaining events that were bound during the execution of the script. These events are activated following the logical web browsing order. For example, a user cannot press a button until the page is loaded; therefore 'onclick' will always follow 'onload'.

To accurately detect which AJAX web applications contain vulnerabilities, Acunetix WVS launches its attacks by sifting through the code (as explained above) in reality emulating a hacker. Hackers do not stop at simply observing the code but would actually execute it even going as far as making the 'necessary' modifications to cause misbehaviour. The efficiency of an Acunetix vulnerability scan is based upon the same techniques which a hacker uses to manipulate the AJAX structure for Cross-site Scripting (XSS), SQL injection, traffic monitoring, session interception, and others.

AJAX-based web applications are rendered in a web browser in modular style by which various single elements of and events in a web page can be loaded and refreshed individually. Thus such applications present a wider window of opportunity for data insertion and manipulation. A thorough audit with Acunetix WVS significantly reduces such associated risks because CSA will parse and execute all scripts. Developers are presented with a more accurate representation of the vulnerabilities contained in their code and may then proceed to make the necessary changes and fixes.

Without the correct interpretation and execution of JavaScript in AJAX applications, a web vulnerability scanner will only limit itself to testing a web application in a global manner instead of its individual components.

3. Summary and Conclusions

The evolution of web technologies is heading in a direction which allows web applications to be increasingly efficient, responsive and interactive. Such progress, however, also increases the threats which businesses and web developers face on a daily basis.

With public ports 80 (HTTP) and 443 (HTTPS) always open to allow dynamic content delivery and exchange, websites are at a constant risk to data theft and defacement, unless they are audited regularly with a reliable web application scanner. As the complexity of technology increases, website weaknesses become more evident and vulnerabilities more grave.

The advent of AJAX applications has raised considerable security issues due to a broadened threat window brought about by the very same technologies and complexities developed. With an increase in script execution and information exchanged in server/client requests and responses, hackers have greater opportunity to steal data thereby costing organizations thousands of dollars in lost revenue, severe fines, diminished customer trust and substantial damage to your organization's reputation and credibility.

The only solution for effective and efficient security auditing is a vulnerability scanner which automates the crawling of websites to identify weaknesses. However, without an engine that parses and executes JavaScript, such crawling is inaccurate and gives website owners a false sense of security.

The Acunetix Client Script Analyzer included in Acunetix WVS, identifies the document object model, events and behaviours of a website and executes all the embedded scripts. This unique web auditing methodology significantly enhances the quality of the vulnerability scan.

Further Reading

FOSTER, J., (2005), “Black Holes: Emerging Web App Security Devices and Products bring Source Code Vulnerabilities to Light”, SearchSecurity.com, http://searchsecurity.techtarget.com/tip/1,289483,sid14_gci1101699,00.html

GARRETT, J.J., (2005), “Ajax: A New Approach to Web Applications”, Adaptive Path: http://adaptivepath.com/publications/essays/archives/000385.php

McLELLAN, D., (2005), “Very Dynamic Web Interfaces”, O’Reilly XML.com, http://www.xml.com/pub/a/2005/02/09/xml-http-request.html

O’REILLY, T., (2005), “What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software”, Oreillynet.com, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

About Acunetix

Acunetix was founded to combat the alarming rise in web attacks. Its flagship product, Acunetix Web Vulnerability Scanner, is the result of several years of development by a team of highly experienced security developers. Acunetix is a privately held company with headquarters based in Europe (Malta), a US office in Seattle, Washington and an office in London, UK. For more information about Acunetix, visit: http://www.acunetix.com; http://www.acunetix.de.