AJAX Apps Ripe Targets for JavaScript Hijacking

April 3 2007

A pervasive vulnerability that allows an attacker to take over any Web browser and silently intercept sensitive data input occurs in Web 2.0 settings from Yahoo to ASP .Net to Google, security firm Fortify says.

Fortify Software has documented what the security firm is calling a "pervasive and critical" vulnerability in Web 2.0 applications - specifically, in the ability of an attacker to use a JavaScript vulnerability to steal critical data by emulating unsuspecting users.

The vulnerability - which allows an exploit called JavaScript Hijacking - can be found in the biggest AJAX frameworks out there, including three server-integrated toolkits: Microsoft ASP.Net AJAX (aka Atlas), Google Web Toolkit and xajax - the last of which is an open-source PHP-class library implementation of AJAX.

Client-side libraries that Fortify inspected and found to be vulnerable are the Yahoo UI, Prototype, Script.aculo.us, Dojo, Moo.fx, jQuery, Rico and MochiKit.

Of the AJAX frameworks and client-side libraries Fortify inspected, only DWR 2.0 (Direct Web Remoting 2.0) has mechanisms to prevent JavaScript Hijacking.

That isn't surprising, given that Joe Walker, who developed DWR, wrote about the JavaScript Hijacking flaw in early March.

According to Fortify, the other AJAX frameworks don't explicitly provide any protection, nor do their documentation materials mention the vulnerability as a security concern.

Brian Chess, Fortify Software's co-founder and Chief Scientist, told eWEEK that the security firm is getting a ho-hum reaction from some regarding the news, since JavaScript has never been considered to be safe anyway.

"Everybody hears, 'Oh, there's a JavaScript security problem,' and everybody says, 'Oh yeah, everybody knows JavaScript is a security concern in itself,'" Chess said.

This, however, is the first type of JavaScript problem that Chess knows of that specifically targets AJAX-style and Web 2.0-style applications, he said.

What's happening, Chess said, is that AJAX-style applications are dropping the X off of AJAX, which stands for Asynchronous JavaScript and XML. Thus, the applications are doing all their work in JavaScript, particularly using it as their data transport format.

The gotcha is that Web browsers don't protect JavaScript as they do HTML or other protocols they transport. This allows rogue hackers to get hold of sensitive data that most developers think they've protected, Chess said.

"The attacker can put code in a Web page," he said. "If he can trick you into running it in your browser, your browser can look like you and act like you, but it's not you; it's actually shoveling data back to - the attacker - ."

The problem specifically lies in JSON (JavaScript Object Notation), a

lightweight data interchange format that for some time has been known to have security problems.

The text-based, human-readable format for representing objects and other data structures is mostly used to transmit structured data over a network connection.

Yahoo began offering some of its Web Services optionally in JSON in December 2005, and Google started offering JSON feeds for its GData Web protocol in December 2006.

One problem with JSON is that CSRF (cross-site request forgery) allows attackers to bypass the technology's cookie-based authentication, as DRW's creator, Walker, says in his blog.

Specifically, CSRF allows a user to invoke cookie-protected actions on a remote server, thus allowing "Mr. Evil to trick Mrs. Innocent into transferring money from her bank account into his," Walker wrote.

Walker is a developer and runs a consultancy called Getahead.

"I believe that JSON is unsafe for anything but public data unless you are using unpredictable URLs," he said in the same blog posting.

Walker said that another, less well-known flaw in JSON is an Array hack that allows malicious users to steal JSON data on Mozilla and any other platform with a modern JavaScript interpreter. This in fact is the subject of Fortify's recent work. Fortify's paper can be downloaded here.

"JSON makes JavaScript Hijacking easier by the fact that a JSON array stands on its own as a valid JavaScript statement. Since arrays are a natural form for communicating lists, they are commonly used wherever an application needs to communicate multiple values. Put another way, a

JSON array is directly vulnerable to JavaScript Hijacking. A JSON object is only vulnerable if it is wrapped in some other JavaScript construct that stands on its own as a valid JavaScript statement."

And more details from the Fortify paper:

"Web browsers enforce the Same Origin Policy in order to protect users from malicious Web sites. The Same Origin Policy requires that, in order for JavaScript to access the contents of a Web page, both the JavaScript and the Web page must originate from the same domain. Without the Same Origin Policy, a malicious Web site could serve up JavaScript that loads sensitive information from other Web sites using a client's credentials, culls through it, and communicates it back to the attacker.

"JavaScript Hijacking allows an attacker to bypass the Same Origin Policy in the case that a Web application uses JavaScript to communicate confidential information. The loophole in the Same Origin Policy is that it allows JavaScript from any Web site to be included and executed in the context of any other Web site.

"Even though a malicious site cannot directly examine any data loaded from a vulnerable site on the client, it can still take advantage of this loophole by setting up an environment that allows it to witness the execution of the JavaScript and any relevant side effects it may have. Since many Web 2.0 applications use JavaScript as a data transport mechanism, they are often vulnerable while traditional Web applications are not."

Anywhere this vulnerability can occur, it does occur, Chess said, with the exception of in DWR. As for the major companies behind frameworks, most all said they will work on the vulnerability and that it will be fixed in the next version.

Microsoft, for one, told eWEEK that its MSRC is on this and that the company is investigating new public reports of possible vulnerabilities that occur in applications developed using the downloadable Microsoft ASP.NET AJAX framework.

A Microsoft spokesperson said that the company is not aware of any attacks attempting to use the reported issue or of customer impact at this time. Yahoo had not been able to provide comment by the time this story posted.

Google, for its part, has posted [an article](#) that shows developers how to prevent the vulnerabilities described by Fortify in all versions of the Google Web Toolkit.

"We plan to add additional, automatic safeguards in the next version of GWT, due out in the coming weeks, to supplement the security measures developers take on their own," a Google spokesperson added.

These companies have been through the security flaw grinder and know better than to ignore vulnerabilities, Chess said. The problem is that many developers aren't using frameworks from the big players at all - rather, they're rolling their own. Unfortunately, many such developers haven't yet embraced security as their responsibility - and it's this that's prompted Fortify to start banging the drum on the issue.

"Most people don't know when they use these AJAX-style components - i.e., frameworks - that they're at more risk," Chess said. "We need to talk to the AJAX community about what the problem is and what they have to do to address it."

The overwhelming reaction Fortify received from framework maintainers was that this vulnerability is a high-priority fix, Chess said.

What's surprising is the few instances in which framework developers said that security wasn't their problem.

"It makes me really mad to think there are developers out there who are fielding code and who expect people who are going to use that code to figure out all the security ramifications," Chess said.

Chess declined to name names, given that he's still working with them, trying to get recalcitrant developers to address the vulnerability.

In order to make the attack succeed, the browser must be tricked into doing something it wasn't intended to do. In some circumstances, this can be done depending on how JavaScript is formatted.

The Array format is fairly commonly used and makes it easy to trick the browser, Chess said. Exploiting the vulnerability is all about having the conditions necessary to abuse the security policy implemented by a given Web browser. Unfortunately, Fortify found those conditions are fulfilled "a surprising amount of time," he said.

The problem with getting developers to accept responsibility for fixing the vulnerability is easy to spot after comparing AJAX hijacking to, say, buffer overflows, Chess said.

The industry has known about buffer overflows for decades and is usually prompt in addressing them. The problem with JavaScript/AJAX/Web 2.0 security flaws is that there has been no strong message going out to software developers regarding security being their responsibility, Chess said.

"With JavaScript, enterprises are still in the early phases of adopting early programming techniques," he said. "We have an opportunity to get in front of the problem. Before it becomes a widespread" insecure

programming practice, he said.

According to Fortify's paper, applications may be vulnerable if they use JavaScript as a data transfer format and if they handle confidential data.

Nobody knows if this vulnerability is currently being used to steal data. That's because if somebody were using it for thievery, it would be undetectable, Chess said: "It very well could be being exploited right now and we wouldn't know it."

As far as how to fix it goes, Fortify's paper gets into the details. In many cases it would take as few as a dozen lines of code. What's of added interest, Chess said, is how the vulnerability came to be in the first place.

"We've got Web 2.0/AJAX kind of guys who want to do things with browsers and HTML and … - they - really weren't designed to do the work," he said. " - They're using - hacks and kludges to make things work. Sometimes that has unforeseen consequences. You get cobbled-together AJAX."

What's needed are standards and protocols and Web browsers that support them, Chess said. The teams at Microsoft and Mozilla that maintain IE and Firefox are where "the rubber hits the road," he said.

"Once they agree something's a standard, it's a standard," he said. There's a lot of people who try to influence them, but it's really they we look to and take cues from."

This vulnerability will likely further motivate standards setting bodies such as the IETF or the W3C, Chess said. Such organizations have often been where Microsoft's and Mozilla's people have come together to determine what will happen with standards and protocols.

"I think this will further motivate them," Chess said. "They've known about problems in this neighborhood. … But I don't think they've understood what a big deal their security decisions would be."

Copyright 2007 by Ziff Davis Media, Distributed by United Press International

Citation: AJAX Apps Ripe Targets for JavaScript Hijacking (2007, April 3) retrieved 26 April 2024 from https://phys.org/news/2007-04-ajax-apps-ripe-javascript-hijacking.html