# Researcher: JavaScript Attacks Get Slicker

April 19 2007

An Arbor Networks researcher at CanSecWest details JavaScript exploits' increasingly sophisticated means of attack and what tools to use to fight them.

Malicious JavaScript is getting smarter. It's now able to fingerprint victims' Web browsers, vulnerable components and accessible CLSIDs, and deliver custom-tailored exploits, according to Dr. Jose Nazario, senior security engineer for Arbor Networks.

Nazario was referring to NeoSploit, a new malware tool he's seen in the wild that carries at least seven distinct exploits to infect a PC with, from which it can choose based on what that PC's weak points are.

"We're seeing a lot of this in the past several months, a lot of malicious JavaScript - at - large," he said. "People are getting more defensive and offensive with JavaScript."

Nazario was speaking at his session on reverse-engineering JavaScript malware on April 18 here at the CanSecWest security show. What he meant by saying that attackers are using JavaScript more defensively is that researchers are increasingly finding exploit code that's using more sophisticated means to obfuscate itself so that security systems won't pick up on exploits.

Malicious JavaScript delivers browser exploits by, for example, using adodb.Stream() or setSlice() objects, often dropping ActiveX/VBScript content in order to download malware onto a system. Obfuscated

JavaScript in its simplest form uses opaque code to thwart static code review and thereby hide its author's methods and intents. The obfuscation can range from simple ASCII chr() or ord() techniques to Base64 encoding or use of a tool such as iWebTool HTML Encrypt to hide HTML or to string splits or joins to build an AJAX object.

On the simple end, for example, a string split would render a word like this:

```
Daxhi="A"+pplica"+"tion";

Vvu=".";
```

This resulting malicious code, of which the above is a small portion, would be easily detected by a human, but, as Nazario pointed out, it presents an effective stumbling block for automated detection.

Attackers also use double obfuscation, Nazario said. On top of simple joins or splits or single encoding, they use double encoding, often with a custom decoder. While several people like to use the browser to decode such exploits, Nazario said it's a bad idea, as the technique is too slow to get full information from a browser under zero-day conditions.

A better idea, he said, is to divorce the JavaScript engine from the browser with a tool such as NJS - an independent implementation of the JavaScript language developed by Netscape and standardized by the ECMA, and designed to be re-entrant, extensible, fast and programmable. Other decoding tools include SpiderMonkey, a JavaScript-C engine from the Mozilla Foundation, and Rhino, an open-source implementation of JavaScript written entirely in Java.

As Nazario described it, reverse-engineering double-decode malware essentially entails cleaning up the HTML and decoding on the command line. This results in code that still requires decoding, so the process is to repeat until the code is no longer encoded.

However, "Life isn't always this easy," Nazario said. "There are lots of defensive JavaScript - code samples - coming around that kill all sorts of inspection routines."

For example, because NJS doesn't know about "arguments," attackers have used "arguments.callee" to make their code tamper-proof. Callee is a property of arguments as a local variable available within all function objects that allows anonymous functions to refer to themselves. That's necessary for writing recursive strings, which, when executed, cause an endless loop and throw a monkey wrench into reverse engineering.

Enter SpiderMonkey, Nazario said. Because SpiderMonkey will choke on certain functions such as alert() or print(), it's impervious to attackers' use of those functions to mess with decoding efforts. Once it chokes, a researcher doing reverse engineering can use another language, such as Python.

In short, as attackers increasingly use JavaScript as a delivery vehicle for malware, Nazario suggested, one of the first steps in defense should be to learn JavaScript and to learn how to love it - not a hard task, given that it's a relatively easy language, he said.

On the up side, attackers using JavaScript are currently limited by the fact that the language has to be decoded to be used by a browser. Also, the tools attackers are using to obfuscate their code and its intent are primitive. But, given that you need a human to analyze the code, those primitive tools are still effective, Nazario said.

At any rate, JavaScript malware is just taking its baby steps. "They'll continue to push the envelope," Nazario said - in other words, we can likely look forward to seeing Malware 2.0 rise to meet the much-ballyhooed Web 2.0.

Citation: Researcher: JavaScript Attacks Get Slicker (2007, April 19) retrieved 2 May 2024 from https://phys.org/news/2007-04-javascript-slicker.html