

Re: More bollocks from biggots

Source: <http://unix.derkeiler.com/Newsgroups/comp.os.vms/2007-04/msg00183.html>

- *From:* "Richard Maher" <maher_rj@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 4 Apr 2007 06:54:56 +0800
-

Hi JF,

The way I understood it:

Yeh, me too; but I think "understood" is too strong a word for what we're feeling :-)

When that response is executed (eval),

So if you choose to visit sites that execute JavaScript from "Who knows where?" and then use a crappy fudgeware protocol that says something like "He's going to the same domain/realm/box-of-fluffy-ducks" or "I'll send this solid-as-a-rock Session-ID thingy (that any JavaScript can see)" and wrap this poisonous JS in the warm embrace of an erstwhile, context-devoid, token effort at authentication, then we're in the pooh! I might have paraphrased a little bit, but is that the nub of the issue? If so, I can see one or two solutions jumping out at me already.

If you access a malicious web site, it can use the <script> command to bypass

the "request must come from same site", and make the same HTTP requests to Google as the Google Mail HTML/Javascript does.

Agreed! HTTP, in particular the XMLHttpRequest API (and it goes without saying, the inherent tokenism (in all senses of the word) approach to web authentication) is crap!

The results are stored in memory, can then be executed and the malicious javascript then has access

Re: More bollocks from biggots

to the

variables containing your personal data.

OK, but my question/assertion was that once you've left this suspect web-site, that you probably never should've been at, the malicious reach of the injected JavaScript has come to an end. Are you, or the document/anyone, saying this is not the case? Has JavaScript been given TSR functionality that allows it to spoof the Object constructor on any page until Browser Close or Re-boot? I think not.

In essence, javascript's ability to load objects into memory without displaying/executing them allows malicious code to execute and transfer information between 2 sites without you knowing about it.

IMHO, the core issue is one of not wanting the user to have to re-enter their credentials (username/password) for *every* message in is this context-devoid, piss-poor excuse for a middleware protocol! This is the poisonous tree from which no edible fruit will ever grow!

Cheers Richard Maher

PS. I'm almost finished (adding floating <DIV>s with real-time queue status info, and predictive text on the queue name) to this DEMO VMS Queue lookup program. I would be very grateful if you could try it out and hack it for me. It's honestly too sexy for words! (And you won't believe how easy the VMS COBOL and HTML/JavaScript is!)

"JF Mezei" <jfmezei.spamnot@xxxxxxxxxxxxxxx> wrote in message [news:9ce7d\\$46121911\\$cef8887a\\$32169@xxxxxxxxxxxxxxx](mailto:news:9ce7d46121911cef8887a$32169@xxxxxxxxxxxxxxx)

Richar Maher wrote:

>For anyone else who has the time to look this up (it's about five pages
>before they start to tell you what they're talking about :-()) can you

please

>give me an english version of the exact vulnerability scenario?

http://www.fortifysoftware.com/servlet/downloads/public/JavaScript_Hijacking.pdf

The way I understood it:

Re: More bollocks from biggots

Say you are a legitimate member of Google Mail. They use fancy javascript which

dynamically generates its own HTTP requests. These requests result in a javascript response stored in memory. When that response is executed

(eval), it

does whatever Google wanted it to do, for instance, setting an array that contains all your email contacts.

If you access a malicious web site, it can use the <script> command to bypass

the "request must come from same site", and make the same HTTP requests to Google as the Google Mail HTML/Javascript does. The results are stored in memory, can then be executed and the malicious javascript then has access

to the

variables containing your personal data. That malicious javascript can

then make

its own covert HTTP request that contains, as parameters, the data it

obtained

from loading the Google javascript response.

In essence, javascript's ability to load objects into memory without displaying/executing them allows malicious code to execute and transfer information between 2 sites without you knowing about it.