

Re: [Full-disclosure] URI handling woes in Acrobat Reader, Netscape, Miranda, Skype

Source: <http://www.derkeiler.com/Mailing-Lists/Full-Disclosure/2007-10/msg00196.html>

- *From:* "john lokka" <merigoth@xxxxxxxxxx>
 - *Date:* Tue, 9 Oct 2007 14:09:44 -1000
-

I know I've jumped on this conversation late, but it seems to me that this is not a "URI/RL" handling problem. Like was mentioned earlier, this is probably a content-type/file association/command string handling problem. I was recently researching the image file "exploit" from splitbrain.org (http://www.splitbrain.org/blog/2007-02/12-internet_explorer_facilitates_cross_site_scripting). During the research, I ran across the IE7's dev team's blog (<http://blogs.msdn.com/ie/archive/2005/02/01/364581.aspx>) where they describe how IE7 determines how to process content based upon the first 256bytes of the information rather than directly based upon how information is passed from the server. I quote "If it cannot find the clsid, the file will be Shell Executed<<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/functions/shell> will use the extension to determine the application to handle the file." So, I wonder if these two incidents are some how related at the shell process level.

In the original posting, The string ([mailto:test%..../..../windows/system32/calc.exe".cmd](mailto:test%..../..../windows/system32/calc.exe)<test%..../..../windows/system32/calc.exe".cmd>) would be processed as a command script as indicated by the percent sign (variable identifier) and the ".cmd" and "overwrite" the mailto initiator. In the case of the ".doc" and ".txt", it appears the final association will determine how the rest of the string is processed.

File association plays a role because command line will attempt to launch the default mail handler. This seems to indicate the problem is at a "lower" level than the browser/mail client. This would explain why it works equally well despite having Firefox, IE7, thunderbird, or outlook.

Based upon the IE7 blog posting, it seems this fundamental change may have been introduced with IE7/XPSP2. However, this is mainly theory. I have tested the "malicious" string on an XP (no service pack) and IE6. The string was processed as described. Meaning the default mail-client (in this case outlook express6) launched with the string in the "TO:" line of a blank email.

All of this may seem obvious, and if I'm re-stating information, I apologize. As far as possible mitigation techniques, microsoft also described the "new functionality" of XP SP2 in this tech net article,

Re: [Full-disclosure] URI handling woes in Acrobat Reader, Netscape, Miranda, Skype

<http://technet.microsoft.com/en-us/library/bb457150.aspx>. They describe what registry settings can affect how content logic can be forced. I haven't really tested this solution because I've been busy. But these are some loose thoughts on the subject.

merigoth

On 10/7/07, gjgowey@xxxxxxxxxxxxxxxxxxxxx <gjgowey@xxxxxxxxxxxxxxxxxxxxx> wrote:

I think that you're both right, but the only solution is the same old, same old: speed, code size, and maintainability/complexity versus the padding and added IO checking of a very secure app. Nothing new, nothing different. It's the same problem that has existed since the dawn of programming.

Invariably the answer to the question will be a proposal for yet another secure programming language or framework. So far all attempts at either have seemed to have failed miserably or never really gained traction (anyone else here remember plan 9 and its more secure brother inferno?). Now the industry trend du jour is moving more towards protecting the rest of the system from unforeseeable vulnerabilities and their exploitation (selinux, dep, etc) when things blow up rather than demand that code is 100% bulletproof.

Bad idea? I'm not so sure. The bigger the system, invariably, the harder to debug to absolute stability in a timely fashion, but source code analysis tools can help lighten the load a bit. However, no amount of auditing of your own product can prevent the problem of a buggy third party that, even if you pass it input in the exact fashion as specified by the manufacture, is still vulnerable. The point of this rambling? No amount of hyperventilating over the security of your own code will ever make it absolutely secure as long as you're placing a reliance on the security of a 3rd party library (in which case I hope you're planning to write everything from the bios of the computer up to the app).

Point? Mitigating the threat posed by unknown attack vectors is something that may start at the program level, but I'm highly doubtful that it can completely be accomplished at just the program level alone. A secure operating system or security framework will pretty much always be a necessity for guaranteeing a completely secure platform.

If there's a silver lining here it's that even the most novice computer user knows that security is a problem with computers as opposed to "security? What's that?" (followed by a deer in the headlights look). That widespread knowledge is what drives budgets to spend on security oriented products rather than the old philosophy that those are optional products. Hopefully, that will eventually materialize in the form of better, cheaper source code auditing products that can help fix the problem at where it all starts: insecure code created by innocent oversight of the

Re: [Full-disclosure] URI handling woes in Acrobat Reader, Netscape, Miranda, Skype

Re: [Full-disclosure] URI handling woes in Acrobat Reader, Netscape, Miranda, Skype

programmer who creates it through either it being absolutely complex, a rushed development cycle, or maybe just the infamous ("that looks right").

Geoff

Sent from my BlackBerry wireless handheld.

-----Original Message-----

From: "Geo." <geoincidents@xxxxxxx>

Date: Sun, 7 Oct 2007 22:26:21

To: <bugtraq@xxxxxxxxxxxxxxxxxxxx>, <full-disclosure@xxxxxxxxxxxxxxxxxxxx>

Subject: Re: [Full-disclosure] URI handling woes in Acrobat Reader, Netscape, Miranda, Skype

----- Original Message -----

From: <Valdis.Kletnieks@xxxxxxx>

2) That said program can protect itself against overtly malicious

input.

Ok then, I can mark you down as one who believes that all the php exploits blamed on bad code writing are actually the fault of php and not the application coded using it's powerful functionality?

Geo.

Full-Disclosure – We believe in it.

Charter: <http://lists.grok.org.uk/full-disclosure-charter.html>

Hosted and sponsored by Secunia – <http://secunia.com/>

Full-Disclosure – We believe in it.

Charter: <http://lists.grok.org.uk/full-disclosure-charter.html>

Hosted and sponsored by Secunia – <http://secunia.com/>