

Re: Cross Site Scripting Vulnerabilities – XSS

Source: <http://www.derkeiler.com/Mailing-Lists/securityfocus/pen-test/2002-08/0013.html>

From: Jeremiah Grossman (jeremiah@whitehatsec.com)

Date: 08/06/02

Date: Tue, 06 Aug 2002 13:22:00 -0700
From: Jeremiah Grossman <jeremiah@whitehatsec.com>
To: Jason binger <cisspstudy@yahoo.com>

A procedure for testing for XSS problems is largely variable upon the implementation and design of the system in question. Testing for XSS in Web/HTML mail is different from Message Boards, is different from web server 404 echo testing, its different from CGI input echoing. The successful execution of script in each instance might be similar, but the implementation on how this is achieved may be quite different. This is because the input and output data land in varied places. So currently, there is no formal procedure that I have some across for testing XSS effectively. We all rely on our experience.

Now that I have given you no help...let try something different...

The general idea with regard to XSS testing is that you are trying to input data into a web application which will then execute script when viewing the output (wherever that may be). The emails before have given you good feedback on how they do things, so lets give you a few other ideas.

Web Application should never store or worse output "<" and ">" characters without converting them into their equivalent HTML entities. If they do, for the sake of ALLOWING HTML (example: WebMail), the web application must endure the task of separating Client-Side Script from HTML. Something I have never seen a web application effectively perform. As I have said in the past, if you allow HTML to be output, your allowing script to hit the client-side (until someone demonstrates otherwise.)

Upon testing CGI's, here's the procedure I take. Identify each web application, document every input source, what's it used for and where the output lands. I then proceed to input the simplest of data strings, such as "<" or ">" and view the output source to see what the filters (if any) did. If I am able to get even an innocuous "<U>" tag through to the output, it's a problem that needs to be corrected. Remember, input is not just limited to input CGI parameters, but also one must consider HTTP Headers as well or anything a web app may use that originates on the client-side. Now this might have helped a little.

SecurityFocus Penetration: Re: Cross Site Scripting Vulnerabilities – XSS

Lets confuse the situation. There are many techniques that have been used to bypass filters designed to thwart XSS. You may have heard terms such as "URL Encoded Strings", "Filter Bypass Manipulation", etc. There are cheat sheets (like Bill's) many professionals use when testing security measures that prevent XSS. All these different XSS variants must be test ed thoroughly, because one hole ruins all the security work.

```

```

Jeremiah–

Dave Aitel wrote:

```
> I just use SPIKE Proxy, modify each variable by hand to be something
> with a <, and then look at the result to see if it's messed up. Maybe
> I'll throw in something to SPIKE real quick to generate a browser window
> for each variable fuzzed and then you can quickly tab through them and
> be done with it.
>
> –dave
> "Cross site scripting is a problem that affects us all." – The unix
> terrorist, GOBBLES talk, DefCon 0x0a
>
```

This list is provided by the SecurityFocus Security Intelligence Alert (SIA) Service. For more information on SecurityFocus' SIA service which automatically alerts you to the latest security vulnerabilities please see: <https://alerts.securityfocus.com/>

- **Previous message:** [Dave Aitel: "SPIKE 2.5 and associated vulnerabilities in Microsoft SQL Server and Exchange 2000"](#)
- **Maybe in reply to:** [Jason binger: "Cross Site Scripting Vulnerabilities – XSS"](#)
- **Next in thread:** [Jeremy Junginger: "RE: Cross Site Scripting Vulnerabilities – XSS"](#)
- **Next in thread:** [Bill Pennington: "Re: Cross Site Scripting Vulnerabilities – XSS"](#)
- **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)