Cross-Site Scripting Vulnerabilities

Jason Rafail, CERT® Coordination Center

Have you ever mistyped the address of a web site and received a message like “Error - *page name* could not be found” or “The page you requested: *page name* does not exist”? Certainly you have, and odds are you never gave it a second thought; you simply corrected the address or went to a different site altogether. It happens all the time. There are plenty of dead links, or links with typos to stumble upon. However, when you encounter an error message like the two listed above, you are actually witnessing a potential security breach—not necessarily against the site, but rather against you directly.

Suppose you entered the following valid URL:

> http://www.example.com/FILENAME.html

If the document "FILENAME.html" did not exist, the web site could return an error message such as

> <HTML>
> 404 page does not exist: FILENAME.html
>
> ....
> </HTML>

Notice that "FILENAME.html" is a string that you entered. The web site has included it in the page returned straight through to your browser.

This may seem harmless, but now imagine that you are browsing through auctions on a popular site; let’s call it auctions.example.com. You come across several auctions that someone has posted and would like to see more items that the same person has for sale; let’s assume this person is a “bad guy” (though you don’t know it) and call him BG12345. You click on BG12345’s website and see a listing of his auctions. You click on a link on his page that interests you and are taken to auction.example.com’s site displaying that item. You scroll down to place a bid, and the auction site prompts you for your name and password to sign in. You enter all the information and hit the submit button. Everything looks fine, but in reality, the information that you submit is getting sent back to BG12345. How can this be? The answer is that auction.example.com has what is known as a cross-site scripting (CSS) vulnerability.

A CSS vulnerability is caused by the failure of a site to validate user input before returning it to the client’s web-browser. The essence of cross-site scripting is that an intruder causes a legitimate web server to send a page to a victim's browser that contains malicious script or HTML of the intruder's choosing. The malicious script runs with the privileges of a legitimate script originating from the legitimate web server. The two error messages mentioned earlier could be examples of such a situation. If instead of entering a page name, you entered an HTML or

script tag, the server would have returned that command to your browser, as well. Your browser would assume the HTML or script tag was from auction.example.com. It would run the script with the privileges that are set up for that site, and when you looked at the website, everything would appear to be normal.

BG12345 used the same method to deceive you. When you clicked on the link to BG12345's auction, the link was actually to an invalid page. The link may have looked something like the example below, it used HTML and scripting to mimic the auction site's page exactly. However, when you clicked submit, it used a form that passed your information back to BG12345. Now BG12345 can access your account, place bids, and change your information. BG12345 can also change your password and lock you out of your own account. Even worse, BG12345 can see the credit card number that you registered with.

So what did BG12345 do? BG12345's web site offered a link to auction.example.com that looked something like this:

```
<A HREF=http://auction.example.com/<script>alert('hello')</script>">Click Here</a>
```

The "FILENAME.html" submitted to auction.example.com was,

```
<script>alert('hello')</script>
```

auction.example.com then used its ordinary routines to generate an error page to you that read,

```
<HTML>
404 page not found: <script>alert('hello')</script>

....
</HTML>
```

In effect, BG12345 managed to "inject" a JavaScript program into the page returned to you by auction.example.com. The JavaScript ran as though it originated at auction.example.com, and could therefore process events in that document. It also maintained communication with BG12345 by virtue of scripting that BG12345 put in the link; this is the way a CSS vulnerability can be exploited to "sniff" sensitive data from within a web page, including passwords, credit card numbers, and any other arbitrary information you input. There are a number of variants to this problem. Odds are that bank.example.com also has the same vulnerability somewhere on its site. BG12345 could potentially access your bank account and transfer funds using the same process.

So what can be done?

- The best protection is to disable scripting when it isn't required. However, even this does not prevent the injection of malicious HTML. You should also protect yourself by accessing security sensitive pages directly instead of following links from unknown sources, or untrusted sites. For example, don't trust a link to your banking site that is in an email message. If you need to access your banking site, go there directly. And, as always, exercise caution when supplying personal information.
- Webmasters can also help. They can ensure that none of their pages return user input that has not been validated. They can also encourage users to disable scripting.
- Another solution is to have "signed scripting" such that any script with an invalid or untrusted signature would not be run automatically. Suggestions of this nature, however, would require changes to the current Internet standards and specifications. Such changes would have to be submitted for consideration to the World Wide Web Consortium (www.w3c.org) or the Internet Engineering Task Force (www.ietf.org).
- If you notice an instance of cross-site Scripting notify the webmaster of that site, and cc the CERT Coordination Center.

Unfortunately, security is often sacrificed in favor of functionality. But, if you browse the Internet with scripting enabled, there is very little you can do to protect your personal information. Cross-site scripting is easy to overlook, and simple to correct. However, it can cause significant damage–your passwords and credit card numbers can be unknowingly divulged to untrusted sources.

Appendix: References and additional information

http://www.cert.org/advisories/CA-2000-02.html

http://www.cert.org/tech_tips/malicious_code_mitigation.html

http://www.kb.cert.org/vuls/id/672683

http://www.kb.cert.org/vuls/id/642239

http://www.kb.cert.org/vuls/id/560659