Protect your apps from cross-site scripting (XSS) attacks

Using escape sequences in HTML output

Doug Tidwell
April 14, 2017

One of the most common techniques for cross-site scripting attacks is injecting code into a web page. This code snippet shows you how to use escape sequences so that any injected code can't run.

Try it yourself!

In this interactive tutorial, you can run the code, try changing it, run it again, and see the results.

One of the most common techniques for cross-site scripting attacks is injecting malicious code into a web page. The code snippet in this short tutorial shows you how to use escape sequences so that any injected code can't run.

Note: This tutorial is an abbreviated version of Usha Ladkani's developerWorks tutorial, "Prevent cross-site scripting attacks by encoding HTML responses." See that tutorial for more details.

How a cross-site scripting attack works

In a cross-site scripting (XSS) attack, the attacker injects malicious code into a legitimate web page that then runs a client-side script. When a user visits the infected web page, the script is downloaded to the user's browser.

Suppose the attacker injects this HTML string into a web page:

```html
<script>alert("You've been attacked!")</script>
```

When the browser loads the web page, it runs the script as part of rendering the page. In this case, the script runs, and the user sees an alert pop up that says "You've been attacked!"
Your defense: Encoding HTML in variables in a server-side Java™ application

To ensure that malicious scripting code is not injected into your page, your best line of defense is to encode all variable strings before they're displayed on the page. Encoding merely means converting every potentially dangerous character to an HTML entity.

The HTML string above will look like this when escaped:

Listing 1. Escaped HTML

```html
&lt;script&gt;alert("You've been attacked!");&lt;/script&gt;
```

Your turn! Try changing the code in Listing 1

Click Run to execute the code as is. You should see the same output as in Listing 1. Then change the variable unescapedText to see what happens. (Keep in mind that you'll need to escape any double quotes with a backslash: `\"`).

```java
import java.lang.StringBuilder;
import java.util.HashMap;

public class EscapeUtils {

    public static void main(String[] args) {
        String unescapedText = "&lt;script&gt;alert("You've been attacked!");&lt;/script&gt;";
        System.out.println(escapeHtml(unescapedText));
    }

    public static String escapeHtml(String inputString) {
        StringBuilder builder = new StringBuilder();
        char[] charArray = inputString.toCharArray();
        for (char nextChar: charArray) {
            String entityName = charMap.get((int) nextChar);
            if (entityName == null) {
                if (nextChar > 0x7F)
                    builder.append("&#").append(Integer.toString(nextChar, 10)).append(";");
                else
                    builder.append(nextChar);
            } else
                builder.append(entityName);
        } return builder.toString();
    }

    public static final HashMap<Integer, String> charMap =
    new HashMap<Integer, String>();

    static {
        charMap.put(34, "\""); // double quote
        charMap.put(35, "\#35\""); // hash mark (no HTML named entity)
        charMap.put(38, "\&\""); // ampersand
        charMap.put(39, "\'\""); // apostrophe, aka single quote
        charMap.put(60, "\<&\""); // less than
        charMap.put(62, "\">\""); // greater than
    }
}
```
Summary

Cross-site scripting is still one of the most common ways to attack a user's machine. However, you can largely block an attacker's ability to infect your web application with malicious code. When writing your application, be sure to encode all variable output in a page before sending it to the user's browser.

One more thing

Be aware that the Java String.toCharArray() method will corrupt strings that contain emojis and other special characters. If you'd really like to dive in to this topic, start with the Wikipedia article on the Unicode Basic Multilingual Plane.
Related topics

- Prevent cross-site scripting attacks by encoding HTML responses
- Examine different types of cross-site scripting attacks

© Copyright IBM Corporation 2017
Trademarks
(www.ibm.com/developerworks.ibm/trademarks/)