A Firefox Extension for Detecting Stored Cross Site Scripting Attack on a Webpage

Tanusha Nadkarni

Department of Computer Engineering
National Institute of Technology Karnataka, Surathkal

Abstract
Cross Site Scripting (XSS) attacks are a form of code injection attacks. It involves injection of HTML tags into user fields provided by websites which are then sent to a user’s web browser. In the browser, Javascript is executed and used to transfer sensitive data to the attacker. In this paper, a way for detection of stored cross site scripting present on a webpage is proposed as an extension for Mozilla Firefox. It works by checking all the Javascript present on a webpage against a blacklist of stored XSS vectors.

1 Introduction
XSS tops the top 10 vulnerabilities list by Open Web Application Security Project (OWASP). The three types of XSS attacks are DOM based XSS attacks, Reflective XSS attacks and Stored XSS attacks. DOM based XSS attacks occur when JavaScript uses input data or data from the server to write dynamic HTML (DOM) elements. In Reflective XSS attacks, code is reflected back to victim. A user is tricked into clicking on a malicious link or submitting a specially crafted form, the injected code travels to the vulnerable web server, which reflects the attack back to the user’s browser.

Most modern web sites such as social networking sites and blogging sites allow users to post content in the form of posts, comments, scraps etc. If this content published by users contains Javascript, then visitors to the site can be exposed to cross-site scripting (XSS) attack.

According to CERT coordination center, a web page contains both text and HTML markup that is generated by the server and interpreted by the client browser. Web sites that generate only static pages are able to have complete control over how the client browser interprets these pages. Whereas web sites that generate dynamic pages do not have complete control over how the client browser interprets these pages. If untrusted content can be introduced into a dynamic page, neither the server nor the client has enough information to detect this malicious activity. In this paper we propose an extension for Firefox browser to detect stored XSS attacks.

Section 2 describes Stored XSS attack along with examples. Section 3 demonstrates how stored XSS can be used for stealing cookies. In section 4 we have provided details of available Firefox
Extensions. Our proposed Firefox extension is discussed in section 5.

2 Stored XSS Attack

Stored XSS attacks (also known as persistent attacks) are those where the scripts are permanently stored on the target servers, e.g. in a database. It is called persistent because it will occur till the message containing malicious script is not deleted. Examples include blogs or forums where users can post content that will be displayed to other users. A user posts a comment in a blog and embeds some JavaScript. The result is that every web browser that renders this comment of the blog will also retrieve the malicious script from the server and execute the attacker’s JavaScript. The attacker’s code can steal the user’s cookies and thus, hijack user’s session. Fig 1 shows the Javascript `<script>alert('XSS')</script>` posted in blog post. When the blog post is viewed the alert is generated as shown in Fig 2.

3 Stealing cookies using XSS

A cookie can be retrieved from the browser with the script `document.cookie` [1]. Blogs hosted by blogspot.com are vulnerable to XSS attacks. For purpose of cookie stealing a blog hosted by blogspot.com was used and the following XSS vector was used


- This script was posted in a blog http://labxssproj.blogspot.com hosted by blogspot.com.
- 172.16.16.106 is the IP address of the machine hosting insert.php and the database storing cookie values.
- Whenever any user visits http://labxssproj.blogspot.com, they will get redirected to `http://172.16.16.10/labproj/insert.php/?vector=\'+document.cookie, and their cookie will be saved in the database of the machine 172.16.16.106. This cookie can then be used for session hijacking.

![Figure 1: Javascript posted](image1.png)

![Figure 2: Javascript is executed and alert is generated](image2.png)
4 Available Mozilla Firefox Extensions

4.1 YesScript Firefox Extension
YesScript [3] lets you make a blacklist of sites that are not allowed to run JavaScript. YesScript does not improve user security. Its only use is on sites that annoy you and consume your system resources.

4.2 noXSS Firefox Extension
noXSS [4] is a Firefox extension that protects against reflective XSS. Basically noXSS checks all executed scripts against relevant request data. If a certain amount of request data is found within a script noXSS assumes that a XSS attempt has occurred and prevents the execution of the whole script.

4.3 NoScript Firefox Extension
The NoScript Firefox extension [5] allows JavaScript, Java and Flash and other plugins to be executed only by trusted web sites of user’s choice. NoScript's unique whitelist based pre-emptive script blocking approach prevents exploitation of security vulnerabilities with no loss of functionality. NoScript provides protection against protection against DOM XSS and Reflective XSS. Scripts (and other blockable elements) are allowed or blocked based on the source from where the script is fetched. Many webpages fetch elements such as iframes, style sheets, scripts and embeddable objects from remote sites. When a webpage includes scripts and other blockable elements from many sources, the user may specify blocking policy for the main address and each of the sources separately.

5 Proposed Extension
For stored XSS detection an extension for Mozilla Firefox called ‘StoredXSSdetector’ was created. This extension maintains a blacklist of XSS vectors and commonly used strings of characters in XSS attacks such as document.cookie. The Blacklist is stored in a database. After a page is rendered in browser, the extension will check for all the script tags in the browser.

Using DOM properties of HTML, the extension checks if the content of <script> elements contain any of the XSS vectors given in database. The getElementsByTagName() returns all elements with a specified tag name. This node access method can be used for finding all the <script> tags in the web page. The easiest way to get or modify the content of an element is by using the innerHTML property. It can also be used to view the source of a page that has been dynamically modified. The innerHTML property can be used to view contents of all script tags and check them against the blacklist of XSS vectors. The extension generates alerts warning user of the XSS vector present in the rendered webpage, hence making the user aware of the XSS attack.

Acknowledgement
I would take this opportunity to express my deep sense of gratitude to Mr. Alwyn Roshan Pais (Assistant Professor, Computer Engineering Department, N.I.T.K, Surathkal) for his immense support and valuable guidance.

References

