SECURING APACHE : XSS INJECTIONS - III

Latest trends in XSS

Meta-information XSS (miXSS)

This new type of XSS vulnerability has emerged recently, and exploits commonly used network administration utilities. It is found in those services that utilise valid user-provided input to gather data and display it for the user. It is in this data that the cross-site scripting occurs. Attackers can extract information about network administration utilities. Websites that allow you to perform DNS resolution, and websites that verify SPF records are more vulnerable to miXSS attacks. To learn more about miXSS, check the resources at the end of this article.

XSS Shell

The XSS Shell is a tool that can be used to set up an XSS channel between a victim and an attacker, so that an attacker can control a victim's browser, sending it commands. The communication is bi-directional.

XSS Tunnel

This is a GPL-licensed open source application written in .NET, and is the standard HTTP proxy which sits on an attacker's system. It enables tunnelling of HTTP traffic through an XSS channel, to use virtually any application that supports HTTP proxies.

XSS causing DDoS attacks

Recent trends have seen attackers using XSS-vulnerable sites as an initiator step in performing DDoS (Distributed Denial of Service) attacks. They trick users into downloading and installing plugins. When the user clicks the download link, besides the plugin, a worm or bot (in most cases) gets installed in the background. This worm/bot can give the attacker full privileges over the user's system, and the attacker can then use it to perform DoS attacks, or to spread the botnet.

Botnet: A botnet is an alliance of interconnected computers infected with some malicious software agent (called a bot). Bots are commanded by an operator, and can typically be ordered to send spam mails, harvest information such as license keys or banking data on compromised machines; or launch distributed denial-of-service (DDoS) attacks against arbitrary targets.

Securing a server against XSS

Cross-site scripting is a potential risk for most Web servers and browsers. Attackers are constantly coming up with new types of this attack, but the following best practices can help you secure your system against attackers.

For clients or users

Take a serious and suspicious view of emails or spam mail that contain big, bulky and suspicious URLs. Don't click such links, even if they are to known and trusted sites. Many of these messages try various tricks to coax you into clicking the link. Some of these include offers to make you financially strong and independent; others threaten that you will lose your account on a (legitimate) website unless you "confirm your username and password immediately". Think hard and deep before you click such links, especially with the knowledge you have gained from this article. Obviously, exercise the same level of caution on online message boards and social networking sites as well.

Recent versions of Mozilla Firefox display good security features. For example, Firefox automatically encodes `<` and `>` (into `%3C` and `%3E`, respectively) in the `document.URL` property, when the URL is not directly typed into the address bar. Therefore, it is not vulnerable to DOM-based attacks. For additional security, install browser addons (extensions) such as NoScript, FlashBlock, and the Netcraft toolbar.

You could also try using the Google Chrome browser, which is released with integrated XSS protection.
If you run into a doubtful link that you still want to open, if you don't use Firefox with NoScript, you should disable JavaScript, Java (and Active X, if you're on Windows) before you click the link. Alternately, visit the website by typing its address directly into your browser.

If a link is to a URL-shortening service like "tiny", "tinyurl", "bit.ly", "is.gd", "shorturl", "snipurl", etc., be careful when clicking the link. You may even want to install a second browser for "untrusted" sites; in this browser, do not sign in to any of your trusted and valuable sites, but use it to visit suspicious URLs. If there is actually an attack behind the URL, even if successful, it will probably not net the attacker any useful cookies.

For developers

The best way to check your website for vulnerabilities is to run a Web application security scan against a local copy of it. The best FOSS project available for this is Nikto.
The next preferred option is to properly escape all untrustworthy data, based on the HTML context (body, attribute, JavaScript, CSS, or URL) that the data will be placed into. Developers need to include this escaping in their applications. See the OWASP XSS Prevention Cheat Sheetfor more information about data escaping techniques.
Filtering script output can also defeat XSS vulnerabilities by preventing them from being transmitted to users. When filtering dynamic content, select the set of characters that is known to be safe, instead of trying to exclude the set of characters that might be bad. This is preferable because it's unclear whether there could be any other characters or character combinations that can be used to expose other vulnerabilities.

Check all headers, cookies, query strings, form and hidden fields, and all other parameters against tags such as `<script>`, `<object>`, `<applet>`, `<form>`, and `<embed>`. Also, do not echo any input value without validation.
Do not store plain-text or weakly encrypted passwords/contents in a cookie. Merely hashing the password with MD5 is not strong, since they are just 16 bytes in length, and hence can easily be deciphered with a brute-force method.

If possible/practicable, the cookie's authentication credentials should be associated with a source IP address. Discovering the same cookie coming from different IPs should raise a red flag.

If possible, eliminate single sign-on, and apply password re-verification to avoid session takeover attacks. An attack against a site running with single sign-on has a better chance of being executed without the user noticing.

Using free hosting sites is an essential building block in the attackers' scheme. Free hosting sites need to be more vigilant about who uses their services, and enterprises should consider blacklisting suspicious IP addresses. Also, if people host scripts like cookie fetchers, they should be monitored for illegal activities on the site.

Cookies sent over HTTP(S) cannot be accessed by script via `document.cookie` so try sending cookies over HTTPS only. Also, try using the POST method for form submissions, instead of GET.

Tools of the security trade

- Dotdefender is a Web application attack protection tool that blocks attacks that are manifested within the HTTP request logic. It works perfectly for SQL injection, cross-site scripting, and header tampering.
- KeepNI immediately alerts you if there is any malfunction detected on your website. It assures that your site is fully functional all the time.
- Web application firewalls check for malicious input values and modification of read-only parameters and also block requests and filter out parameters. Their biggest benefit is to protect old (legacy) applications that are insecure.

Source : http://www.opensourceforu.com/2010/09/securing-apache-part-2-xss-injections/