Protect Against Cross Site Scripting (XSS) Attacks

Cross Site Scripting (XSS) is a vulnerability in web applications that allows an attacker to inject HTML, typically including JavaScript code, into a web page. XSS results from the intermingling of server code and user input. If user input is not sanitized correctly, it could contain code that runs along with server code in a client's browser. In 2010, XSS was ranked the #2 web application security risk by the Open Web Application Security Project (OWASP) and the #1 software error by the SANS Institute.

There are two primary types of XSS vulnerabilities – ***reflected*** (non-persistent) and ***stored*** (persistent). In a reflected XSS attack, the attacker persuades a victim to click on a specially crafted link that makes a request to a vulnerable web server. This allows the attacker to run arbitrary code in the victim's web browser. In reflected attacks, the attacker must target each victim individually.

In a stored attack, the attacker embeds code in a web page or other data stored on a vulnerable server. The attacker's code will then run in the browser of everyone who visits the compromised web page. Because of the potential to exploit multiple victims with minimal effort, stored attacks are generally considered to be more dangerous than reflected attacks.

XSS attacks range in severity. Typical XSS payloads include redirection to phishing sites, password logging, session theft, malware distribution, browser automation and the ability to pivot onto an internal network to launch additional attacks.

Users

There are steps that users can take to protect themselves from XSS (and other) attacks. In addition to using common sense while surfing (don't click on links sent from unknown sources, close sessions when finished), users should consider the following measures.

- **Restrict Untrusted JavaScript:** Allowing all JavaScript to run opens a user up to XSS attacks. The most effective (but not foolproof) method for a user to prevent XSS attacks is to allow JavaScript to run only if it comes from a domain that the user explicitly trusts. Installing a browser plug-in that implements domain whitelisting, such as NoScript for Firefox, is highly recommended. Internet Explorer users can achieve whitelisting through the configuration of Trusted and Restricted Security Zones.

- **Use Built-In Browser Protections:** Some browsers have begun to incorporate XSS protection inherently. For example, as of version 8, Internet Explorer includes an XSS filter as well as a SmartScreen filter that uses reputation to protect against malicious websites. These extra security measures should be enabled when available.

- **Restrict External Websites from Requesting Internal Resources:** Allowing external websites to force a browser to request internal resources can allow for an attacker to pivot an attack onto a vulnerable internal website. The NoScript plug-in has a feature called the Application Boundary Enforcer (ABE) that can be configured to disallow external websites from requesting internal resources.

- **Maintain Good System Hygiene:** It is important to keep systems and applications up-to-date with updates and patches, protected from malware and securely configured. For more information, read the following NSA guides: Best Practices for Securing a Home Network and Mitigation Monday #2: Defense Against Drive-By Downloads.

Developers

The most effective way to get rid of XSS vulnerabilities is to ensure that developers understand the dangers of XSS attacks and have tools that allow them to create secure web applications without hindering their productivity. The OWASP XSS Prevention Cheat Sheet has a lot of useful information on XSS attacks and how to process user input safely. There are also tools that help developers create secure web applications without much extra work.

Blacklisting vs. Whitelisting: To help mitigate XSS attacks, two basic techniques are used to sanitize data. *Blacklisting* uses a list of known bad data to block illegal content from being executed. *Whitelisting* uses a list of known good data to allow only that content to be executed.

Blacklisting mode is faster to set up, but can be bypassed more easily by a skilled attacker. Whitelisting allows for a much stronger security solution but comes with a steep learning curve. Once mastered, though, whitelisting is very effective at stopping XSS attacks.

- **OWASP Enterprise Security API (ESAPI):** The ESAPI library is an implementation of methods, including whitelisting, that process user input safely. It is available in a number of modern programming languages such as Java EE, PHP, .NET, Cold Fusion, Python and others. The ESAPI library requires the developer to understand which methods are susceptible to XSS attacks, and replace them with safe implementations accordingly.

- **Microsoft AntiXSS Library:** The Microsoft AntiXSS Library can be used to replace existing ASP.NET methods that process user input with new methods that do so safely. The AntiXSS Library uses a whitelisting approach for filtering content. The AntiXSS Library also includes a DLL that can be included in a project and used to hook all potentially unsafe calls, replacing them with safe alternatives.

- **Web Vulnerability Scanners:** There are many tools or services that scan websites for XSS vulnerabilities. These tools or services crawl through websites and check code that takes user input to see if it is susceptible to XSS attacks. These tools may not catch all XSS vulnerabilities, but they may at least find the low hanging fruit.

Client/Server Coordination

Another technique for mitigating XSS attacks that has started to emerge is using coordination between the web application and the client browser to separate user supplied data from web application HTML.

- **Content Security Policy (CSP):** CSP is a proposed client/server technology standard that was first implemented in Firefox version 4. In CSP, the website administrator segregates scripts from the rest of the web site (putting them into a source file) and whitelists the domains that should be trusted by the browser as valid script sources. Any other scripts that the browser encounters should be presumed to have resulted from an XSS attack. The browser takes this server information and uses it to determine whether it will run a given script or not.

Network Administrators

Modifications to desktop configurations and web application code are often outside the network administrator's control. While protecting the enterprise against XSS attacks by relying solely on network devices can be hard, there are a number of technologies that can help.

- **White Trash Squid Web Proxy Plug-in:** WhiteTrash is a plug-in for the *squid* proxy with goals similar to those of NoScript. It uses whitelists to accept scripts only from explicitly trusted domains. Enterprise management of WhiteTrash is easier than NoScript as the whitelist can be managed on a small number of proxies that all enterprise web traffic must pass through.
- **Web Application Firewalls (WAFs):** A WAF is an Intrusion Detection/Prevention technology that specifically looks at and understands Hyper Text Transfer Protocol (HTTP) traffic. WAFs can sit anywhere on the network but need to be able to view the HTTP traffic unencrypted. They can inspect both inbound and outbound HTTP traffic for vulnerabilities and can operate in either blacklist or whitelist mode.