**What is XSS (Cross Site Scripting)**

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users.

**Types of XSS**

While the goal of an XSS attack is always to execute malicious JavaScript in the victim's browser, there are few fundamentally different ways of achieving that goal. XSS attacks are often divided into three types:

* **Persistent XSS**, where the malicious string originates from the website's database.
* **Reflected XSS**, where the malicious string originates from the victim's request.
* **DOM-based XSS**, where the vulnerability is in the client-side code rather than the server-side code.

**Consequences of XSS**

Account Hijacking: hijack legitimate user accounts by stealing their session cookies

Stealing credentials: steal user credentials, instead of their cookies

Sensitive Data: exfiltrate sensitive data (personal identifiable information or cardholder data)

Drive-by Downloads: drop a payload to hook the victim's browser to the attacker's servers

Other attacks: key logger, port scan, web site defacement

**How to prevent XSS**

* Context-sensitive encoding
* HTML Sanitizing Filters
* Security Encoding Library
* Proper and restrictive white-list input validation
* Anti-XSS Library (provides all of the necessary context-encoding methods)
* Explicitly set the charset, both in the HTTP header AND the META tag - avoid UTF-7
* Encoding data output to avoid stored XSS
* HtmlEncode the fields before printing on the webpage < page wont interpret the data as script
* WAF (web application firewall)
* Crossing boundaries policy – even if a user has a cookie that will automatically log them into your site, they should be forced to enter their username and password when they attempt to access any sensitive account information
* HTML escape JSON values in an HTML context and read the data with JSON.parse - ensure returned Content-Type header is application/json and not text/html
* JavaScript Escape before Inserting Untrusted Data into JavaScript Data Values
* Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes
* HTML Escape before Inserting Untrusted Data into HTML Element Content
* Never Insert Untrusted Data Except in Allowed Locations
* CSS Escape and Strictly Validate Before Inserting Untrusted Data into HTML Style Property Values
* URL Escape before Inserting Untrusted Data into HTML URL Parameter Values
* HTML Sanitization - Sanitize HTML Markup with a Library Designed for the Job

**XSS Prevention Rules Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Type** | **Context** | **Code Sample** | **Defense** |
| String | HTML Body | <span>UNTRUSTED DATA</span> | * HTML Entity Encoding |
| String | Safe HTML Attributes | <input type="text" name="fname" value="UNTRUSTED DATA"> | * Aggressive HTML Entity Encoding * Only place untrusted data into a whitelist of safe attributes (listed below). * Strictly validate unsafe attributes such as background, id and name. |
| String | GET Parameter | <a href="/site/search?value=UNTRUSTED DATA">clickme</a> | * [URL Encoding](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet#RULE_.235_-_URL_Escape_Before_Inserting_Untrusted_Data_into_HTML_URL_Parameter_Values) |
| String | Untrusted URL in a SRC or HREF attribute | <a href="UNTRUSTED URL">clickme</a> <iframe src="UNTRUSTED URL" /> | * Canonicalize input * URL Validation * Safe URL verification * Whitelist http and https URL's only ([Avoid the JavaScript Protocol to Open a new Window](https://www.owasp.org/index.php/Avoid_the_JavaScript_Protocol_to_Open_a_new_Window)) * Attribute encoder |
| String | CSS Value | <div style="width: UNTRUSTED DATA;">Selection</div> | * [Strict structural validation](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet#RULE_.234_-_CSS_Escape_And_Strictly_Validate_Before_Inserting_Untrusted_Data_into_HTML_Style_Property_Values) * CSS Hex encoding * Good design of CSS Features |
| String | JavaScript Variable | <script>var currentValue='UNTRUSTED DATA';</script> <script>someFunction('UNTRUSTED DATA');</script> | * Ensure JavaScript variables are quoted * JavaScript Hex Encoding * JavaScript Unicode Encoding * Avoid backslash encoding (\" or \' or \\) |
| HTML | HTML Body | <div>UNTRUSTED HTML</div> | * [HTML Validation (JSoup, AntiSamy, HTML Sanitizer)](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet#RULE_.236_-_Use_an_HTML_Policy_engine_to_validate_or_clean_user-driven_HTML_in_an_outbound_way) |
| String | DOM XSS | <script>document.write("UNTRUSTED INPUT: " + document.location.hash);<script/> | * [DOM based XSS Prevention Cheat Sheet](https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet) |

**Output Encoding Rules Summary**

|  |  |  |
| --- | --- | --- |
| **Encoding Type** | **Encoding Mechanism** | |
| HTML Entity Encoding | Convert & to &amp; Convert < to &lt; Convert > to &gt; Convert " to &quot; Convert ' to &#x27; Convert / to &#x2F; | |
| HTML Attribute Encoding | Except for alphanumeric characters, escape all characters with the HTML Entity &#xHH; format, including spaces. (HH = Hex Value) | |
| URL Encoding | Standard percent encoding, see: <http://www.w3schools.com/tags/ref_urlencode.asp>. URL encoding should only be used to encode parameter values, not the entire URL or path fragments of a URL. | |
| JavaScript Encoding | Except for alphanumeric characters, escape all characters with the \uXXXX unicode escaping format (X = Integer). | |
| CSS Hex Encoding | CSS escaping supports \XX and \XXXXXX. Using a two character escape can cause problems if the next character continues the escape sequence. There are two solutions (a) Add a space after the CSS escape (will be ignored by the CSS parser) (b) use the full amount of CSS escaping possible by zero padding the value. | |
| **Encoding Type** | | **Encoding Mechanism** | |
| HTML Entity Encoding | | Convert & to &amp; Convert < to &lt; Convert > to &gt; Convert " to &quot; Convert ' to &#x27; Convert / to &#x2F; | |
| HTML Attribute Encoding | | Except for alphanumeric characters, escape all characters with the HTML Entity &#xHH; format, including spaces. (HH = Hex Value) | |
| URL Encoding | | Standard percent encoding, see: <http://www.w3schools.com/tags/ref_urlencode.asp>. URL encoding should only be used to encode parameter values, not the entire URL or path fragments of a URL. | |
| JavaScript Encoding | | Except for alphanumeric characters, escape all characters with the \uXXXX unicode escaping format (X = Integer). | |
| CSS Hex Encoding | | CSS escaping supports \XX and \XXXXXX. Using a two character escape can cause problems if the next character continues the escape sequence. There are two solutions (a) Add a space after the CSS escape (will be ignored by the CSS parser) (b) use the full amount of CSS escaping possible by zero padding the value. | |