**Cross-Site Scripting (XSS)**

**Description 1**

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

**An attacker can use** XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. [1]

**Description 2**

Cross-site scripting (XSS) is a security bug that **can affect** websites. If present in your website, this bug **can allow an attacker to** add their own malicious JavaScript code onto the HTML pages displayed to your users. Once executed by the victim's browser, this code could then perform actions such as completely changing the behavior or appearance of the website, stealing private data, or performing actions on behalf of the user. [2] --- (refrence [2] has more examples about XSS)

**Description 3**

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. A cross-site scripting vulnerability may be **used by attackers to** bypass access controls such as the same-origin policy. Their effect may range from a petty nuisance to a significant security risk, depending on the sensitivity of the data handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner.

**Example 1 [1]**

The following JSP code segment reads an employee ID, eid, from an HTTP request and displays it to the user.

**<% String eid = request.getParameter("eid"); %>**

**...**

**Employee ID: <%= eid %>**

The code in this example operates correctly if eid contains only standard alphanumeric text. If eid has a value that includes meta-characters or source code, then the code will be executed by the web browser as it displays the HTTP response.

Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use e-mail or social engineering tricks to lure victims into visiting a link to the URL. When victims click the link, they unwittingly reflect the malicious content through the vulnerable web application back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS.

**Reference**

[1] <https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)>

[2] <https://www.google.com/about/appsecurity/learning/xss/>

[3] <https://en.wikipedia.org/wiki/Cross-site_scripting>