## Cross Site Scripting

**Description**

Cross Site Scripting is also shortly known as XSS.

XSS vulnerabilities target scripts embedded in a page that are executed on the client side i.e. user browser rather then at the server side. These flaws can occur when the application takes untrusted data and send it to the web browser without proper validation.

Attackers can use XSS to execute malicious scripts on the users in this case victim browsers. Since the browser cannot know if the script is trusty or not, the script will be executed, and the attacker can hijack session cookies, deface websites, or redirect the user to an unwanted and malicious websites.

XSS is an attack which allows the attacker to execute the scripts on the victim's browser.

**Implication:**

* Making the use of this security vulnerability, an attacker can inject scripts into the application, can steal session cookies, deface websites, and can run malware on the victim's machines.

**Vulnerable Objects**

* Input Fields
* URLs

**Examples**

1. **http://www.vulnerablesite.com/home?"<script>alert("xss")</script**>

The above script when run on a browser, a message box will be displayed if the site is vulnerable to XSS.

The more serious attack can be done if the attacker wants to display or store session cookie.

2. **http://demo.testfire.net/search.aspx?txtSearch <iframe**> <src = **http://google.com** width = 500 height 500></iframe>

The above script when run, the browser will load an invisible frame pointing to **http://google.com**.

The attack can be made serious by running a malicious script on the browser.

**Recommendations**

1. White Listing input fields
2. Input Output encoding

## How to Prevent Cross-site Scripting (XSS) – Generic Tips

### Step 1: Train and maintain awareness

### Step 2: Don’t trust any user input

### Step 3: Use escaping/encoding

### Step 4: Sanitize HTML

### Step 5: Set the HttpOnly flag

### Step 6: Use a Content Security Policy

### Step 7: Scan regularly