* Example:

An attacker must first find a way to inject malicious code (payload) into a web page that the victim visits.

print "<html>"

print "<h1>Most recent comment</h1>"

print database.latestComment

print "</html>"

The above script simply takes the latest comment from a database and includes it in an HTML page. It assumes that the comment printed out consists of only text and contains no HTML tags or other code. It is vulnerable to XSS, because an attacker could submit a comment that contains a malicious payload

<script>doSomethingEvil();</script>

The web server provides the following HTML code to users that visit this web page:

<html>

<h1>Most recent comment</h1>

**<script>doSomethingEvil();</script>**

</html>

When the page loads in the victim’s browser, the attacker’s malicious script executes. Most often, the victim does not realize it and is unable to prevent such an attack.

* Solution:

To keep yourself safe from XSS, you must sanitize your input. Your application code should never output data received as input directly to the browser without checking it for malicious code. Use an appropriate escaping/encoding technique depending on where user input is to be used: HTML escape, JavaScript escape, CSS escape, URL escape, etc. Use existing libraries for escaping, don’t write your own unless absolutely necessary.