# Cross-Site-Scripting Mitigation C#

Cross-site scripting (XSS) is a type of security vulnerability that allows an attacker to inject malicious scripts into a web page viewed by other users. To mitigate stored XSS attacks in C#, you can use various techniques, such as input validation, encoding, and output sanitization. Here are some examples:

**Input validation**: Validate all user input on the server-side before storing it in the database. You can use regular expressions, input masks, or whitelisting to ensure that the input conforms to the expected format.

```
string userInput = Request.Form["input"];
if (Regex.IsMatch(userInput, "^[a-zA-Z0-9]+$")) {
  // Store the input in the database
} else {
  // Display an error message
}
```

**Encoding**: Encode user input when displaying it back to the user to prevent the browser from interpreting it as HTML or JavaScript code. You can use HTML encoding, URL encoding, or JavaScript encoding, depending on the context.

```
string userInput = GetFromDatabase();
string encodedInput = HttpUtility.HtmlEncode(userInput);
Response.Write(encodedInput);
```

**Output sanitization**: Sanitize user input to remove any potentially malicious code before storing it in the database. You can use a library like the Microsoft AntiXSS library to perform this task.

```
string userInput = Request.Form["input"];
string sanitizedInput = AntiXssEncoder.HtmlEncode(userInput, true);
// Store the sanitized input in the database
```

These techniques can help mitigate stored XSS attacks in your C# code. However, it's important to note that security is a continuous process, and you should always stay up-to-date with the latest security best practices and vulnerabilities.

Thanks by

@black_hawk