Cross-Site Scripting Mitigation .NET

Cross-site scripting (XSS) is a common vulnerability that affects web applications built on the .NET platform. To mitigate stored XSS attacks in .NET, you can use various techniques, such as input validation, output 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 or data annotations to ensure that the input conforms to the expected format.

```
[Required]
[StringLength(50)]
[RegularExpression(@"^[a-zA-Z0-9]+$")]
public string Input { get; set; }

if (ModelState.IsValid)
{
   // Store the input in the database
}
```

Output 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 the HtmlEncode method or Razor's @Html.Raw method to encode the output.

```
@{
   string userInput = GetFromDatabase();
   string encodedInput = HttpUtility.HtmlEncode(userInput);
}

<p>@encodedInput</p>
```

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

```
string userInput = Request.Form["input"];
string sanitizedInput = Microsoft.Security.Application.Encoder.HtmlEncode(userInput);
// Store the sanitized input in the database
```

These techniques can help mitigate stored XSS attacks in your .NET 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. Additionally, you should consider using a Content Security Policy (CSP) to further enhance the security of your web application.

Thanks by **@black_hawk**