**Title: Cross Site Scripting**

Domain: Vulnweb.com

Subdomain: testasp.vulnweb.com

Steps to reproduce:

Step 1: visit http:// <http://testasp.vulnweb.com/>

Step 2: On the top menu you will find a search option.

Step 3: Click on it and you will be prompted with the Search box.

Step 4: Type <script>alert(1)</script>

Step 5: Click Search.

Step 6: It will prompt you with an alert 1.

Step 7: There you can see that your website is vulnerable to cross site scripting.

Impact: Cross-site scripting (XSS) attacks can have **devastating consequences**. When a web application is vulnerable to XSS, an attacker can inject malicious code that gets executed in a user’s browser. Here are some of the impacts:

1. **Data Theft**: Attackers can steal sensitive data, such as user credentials or personal information, by injecting malicious scripts into vulnerable pages.
2. **Session Hijacking**: XSS allows attackers to hijack user sessions, potentially gaining unauthorized access to accounts.
3. **Malware Distribution**: Malicious scripts can be injected into web pages, leading to the distribution of malware to unsuspecting users.
4. **Website Defacement**: Attackers may modify website content, defacing it or spreading harmful messages.
5. **SEO Manipulation**: XSS can affect search engine rankings by injecting spammy content into web pages.

To mitigate these risks, developers should sanitize user input, validate and escape output, and use security libraries to prevent XSS attacks. Regular security audits are essential to identify and address vulnerabilities

**Mitigation:** Cross-Site Scripting (XSS) is a serious vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. Here are some steps to mitigate XSS risks:

1. **Prohibit HTML Code in Inputs**: Prevent users from posting HTML code into form inputs. This straightforward measure helps reduce the risk of XSS.
2. **Validate Inputs**: When accepting form inputs, validate the data to ensure it meets specific criteria. Proper input validation helps prevent malicious input from being processed.
3. **Secure Your Cookies**: Set the HttpOnly flag for cookies. This helps mitigate the effects of a possible XSS vulnerability by preventing client-side scripts from accessing sensitive cookies.
4. **Sanitize Data**: Use HTML sanitizers to clean and parse user input that needs to contain HTML. This ensures that valid tags are preserved while removing any potentially harmful content.

POC including screenshot is included in the report which is attached below









