**Report**

**Task3**

**Url:** <http://testasp.vulnweb.com/>

Sub-domain: <http://testasp.vulnweb.com/Search.asp>

Steps involved:

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

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

Step 3: On clicking it and we will be prompted with the Search box.

Step 4: We can intercept the request in Burp Suite

Step 5: Now we’ve to find different payloads for XSS.

Step 6: We’ve sent the request to the intruder and pasted all the payloads.

Step 7: We’ll try to find a successful payload for XSS.

Step 8: Then, we’ve prepared this report.

[**A7:2017-Cross-Site Scripting XSS**](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)): XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim’s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites

Impacts

* Moderate for reflected and DOM XSS
* Severe for stored XSS, with remote code execution on the victim’s browser, such as stealing credentials, sessions, or delivering malware to the victim.

How to Prevent

Preventing XSS requires separation of untrusted data from active browser content.

This can be achieved by:  
\* Using frameworks that automatically escape XSS by design, such as the latest Ruby on Rails, React JS. Learn the limitations of each framework’s XSS protection and appropriately handle the use cases which are not covered.

\* Escaping untrusted HTTP request data based on the context in the HTML output (body, attribute, JavaScript, CSS, or URL) will resolve Reflected and Stored XSS vulnerabilities. The [OWASP Cheat Sheet ‘XSS Prevention’](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html) has details on the required data escaping techniques.

\* Applying context-sensitive encoding when modifying the browser document on the client side acts against DOM XSS. When this cannot be avoided, similar context sensitive escaping techniques can be applied to browser APIs as described in the [OWASP Cheat Sheet ‘DOM based XSS Prevention’](https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html).

\* Enabling a [Content Security Policy (CSP)](https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP) as a defense-in-depth mitigating control against XSS. It is effective if no other vulnerabilities exist that would allow placing malicious code via local file includes (e.g. path traversal overwrites or vulnerable libraries from permitted content delivery networks).