**Executive Summary**

The purpose of this assessment is to perform a vulnerability scan on a demo website using online scanners like OWASP ZAP to identify and point out security loopholes, missing best security practices.  
  
**Scope of Testing**

The security assessment including testing for security loopholes was performed on the defined scope below.

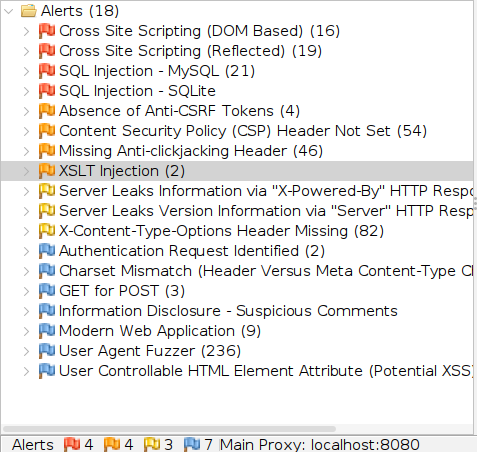
The following was the scope covered under the security audit.

* **Application 1**: <http://testphp.vulnweb.com>

**Scan Results**

There are 18 Alerts which includes the following priority levels

1. High Priority - 4
2. Medium Priority - 4
3. Low Priority - 3
4. Information Priority - 7



**Findings**

1. **Cross Site Scripting**  
   **Severity**: High **Affected Urls**: 19  
     
   **Details of Vulnerability:**

Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client.

**Solution**

To avoid XSS vulnerabilities, use vetted libraries or frameworks that provide constructs for proper encoding, understand data context and required encoding, and encode all non-alphanumeric characters for web page output. Duplicate client-side security checks on the server side, and use structured mechanisms that enforce data and code separation if available. For every web page, specify a character encoding, and consider setting the session cookie to HttpOnly to mitigate XSS attacks. Assume all input is malicious and use an "accept known good" input validation strategy, considering all relevant properties. Perform input validation at well-defined interfaces within the application.

1. **SQL Injection - MySQL  
     
   Severity**: High **Affected Urls**: 21  
     
   **Details of Vulnerability:**SQL injection may be possible. This vulnerability was detected by manipulating the parameter to cause a database error message to be returned and recognised.  
    **Solution**Don't trust client input even if client-side validation is present. Type check all server-side data. Use stored procedures where possible, but avoid dynamic SQL queries using string concatenation. Escape all data received from the client, use allow/deny lists, and apply the principle of least privilege. Grant the minimum database access that is necessary for the application where necessary.
2. **Content Security Policy (CSP) Header Not Set  
     
   Severity**: Medium **Affected Urls**: 54  
     
   **Details:**Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page.  
     
   **Solution**Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.
3. **XSLT Injection**

Severity: Medium Affected Urls: 2  
  
**Details:**Injection using XSL transformations may be possible, and may allow an attacker to read system information, read and write files, or execute arbitrary code. Hence Port scanning may be possible.

**Solution:**  
Sanitize and analyze every user input coming from any client-side.